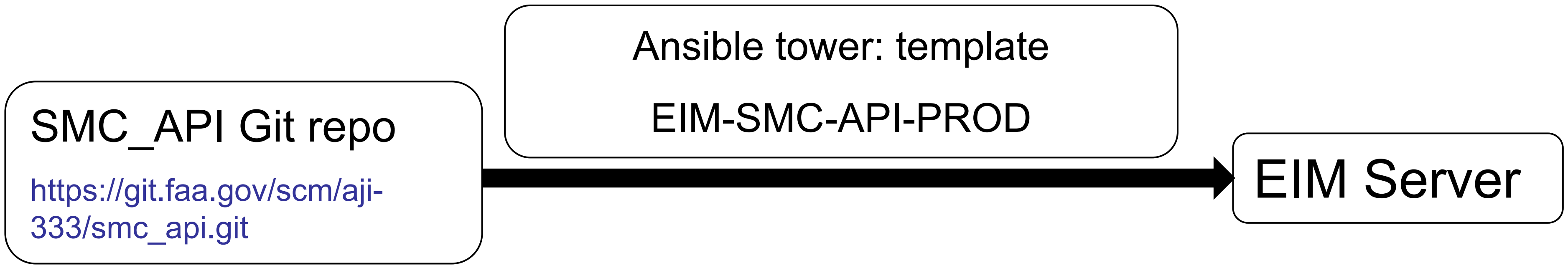
Automation of EIM Deployment of Surface MOR



Details can be found on Teams channel:

TMS-AJO-Data Science and Analytics => SMC

Link:

https://usfaa.sharepoint.com/:b:/r/sites/TMS-AJO-DataScienceandAnalytics/Shared%20Documents/SMC/To_ansiblize_SMC_to_PROD_in_EIM.pdf?csf=1&web=1&e=xMrGM7

Steps during automation of deployment

Major steps in EIM-server when the template in Ansible tower executed:

- Install httpd packages
- Install ssl certificates (gets CA certificates files and edits ssl.conf file as required)
- Ensure firewall https services
- Install dependencies for python (*gcc, openssl, zlib-devel, bzip, libgomp, git, patch, libffi, sqlite, ncurses, xz-devel, libuuid-devel*)
- Install python 3.7
- Create python virtual environment
- Activate virtual environment and run requirements.txt file to install the required python packages
- Create WSGI (Web Server Gateway Interface) directory
- Copy all application codes to WSGI directory in EIM-server from the connected git-repository

Ensemble modeling for surface report classification

Presented to:	BI&A COP
By:	Igor Filippov
Date:	March 1st, 2022

History

- **“Surface MOR Auto Classification”** presented by Firdu Bati in June 2019
- Additional modeling and development by Ramesh Adhikari and Igor Filippov in 2020-2021

Overview

- Surface MOR severity classification:
  Categories A, B, C & D, SI, RE
- SMEs and analysts in the Runway Safety Office manually review approximately 2,500 MORs per year to classify RI, SI, RE events

Original model

- **Feature representation: tf_idf, word embedding (word2vec)**
- **Three Machine Learning Models:**
    - –Gradient Boosting Machine (GBM)
    - –Support Vector Machine (SVM)
    - –Deep Learning –(CNN/RNN/LSTM)

Previous work – 100% support

```
Accuracy Score: 0.8960

Classfication Reprot:
              precision    recall  f1-score   support
         SI      0.917      0.887     0.902      3397
         RE       0.965      0.981     0.973      2299
          D      0.894      0.886     0.890      7143
          C      0.867      0.904     0.885      5578
         AB       0.818      0.076     0.140       118

avg / total      0.898      0.898     0.896     18535
Percentage Support  100%
```

Voting Logic v. 1.0

- 3 Matches: lower threshold (0.75)
- 2 matches : medium threshold (0.85)
- No match: higher threshold (0.95)

Voting Logic v. 2.0

Three match:	
RE, SI	0.8
D	0.85
C, AB	0.95
Two match:	
RE, SI	0.85
D	0.9
C, AB	0.99
No match:	
RE, SI	0.9
D	0.975
C, AB	0.999

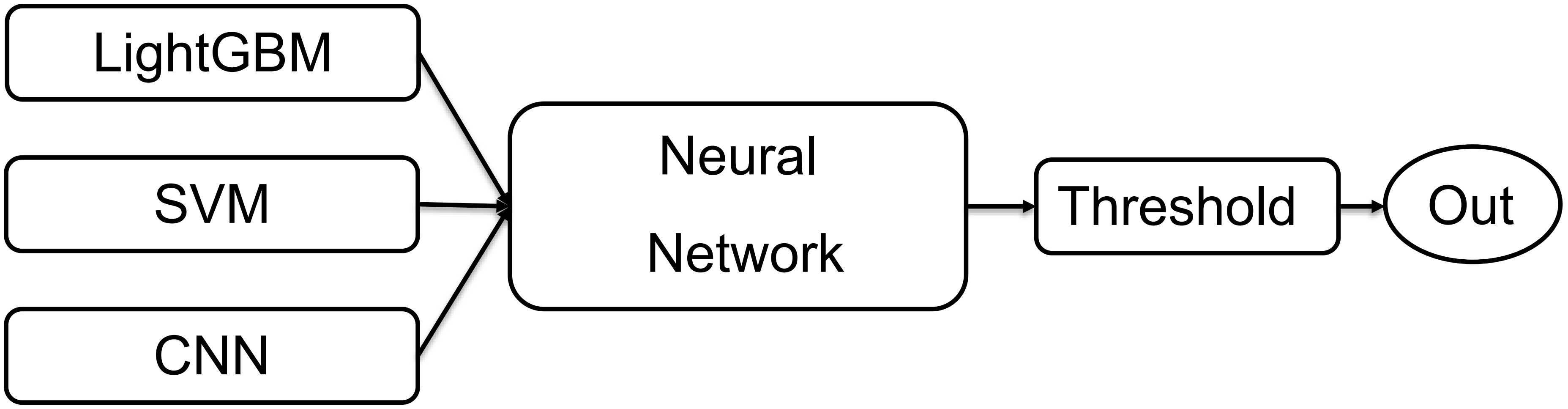
Tested more models

Model1	Model2	Model3	Accuracy	Manual Review (%)	Fractional Support
LGB	XGB	CAT	0.9489	19.46	0.8054
LGB	XGB	CAT	0.9541	21.91	0.7809
CNN	LGB	XGB	0.9241	13.87	0.8613
LGB	XGB		0.9308	15.61	0.8439
CNN-LSTM	SVM	XGB	0.943	18.84	0.8116
CNN-LSTM	LGB	XGB	0.9437	19.02	0.8098
CNN-LSTM	XGB	CAT	0.9455	19.36	0.8064
CNN-BiLSTM	XGB	CAT	0.9472	20.12	0.7988
CNN-BiLSTM	LGB	XGB	0.9457	20.23	0.7977
CAT	XGB		0.9313	15.78	0.8422
CNN	LGB	CAT	0.9243	20.58	0.7942
RF	LGB	XGB	0.959	25.20	0.7480
LGB	SVM	XGB	0.9493	21.33	0.7867
SVM	XGB	CAT	0.9513	21.62	0.7838
CNN	LGB	SVM	0.9252	20.40	0.7960
CAT	LGB		0.9527	25.38	0.7462

Voting – 72% support

```
Accuracy_score: 0.9626217228464442

Classification Reports:
             precision    recall  f1-score   support
        SI       0.945     0.961     0.953      2917
        RE        0.973     0.992     0.982      2239
         D       0.967     0.951     0.959      4373
         C       0.966     0.974     0.970      3764
        AB        0.000     0.000     0.000        57

avg / total       0.959     0.963     0.961     13350
Percentage Support: 72.0%
```

Ensemble model



Neural network architecture

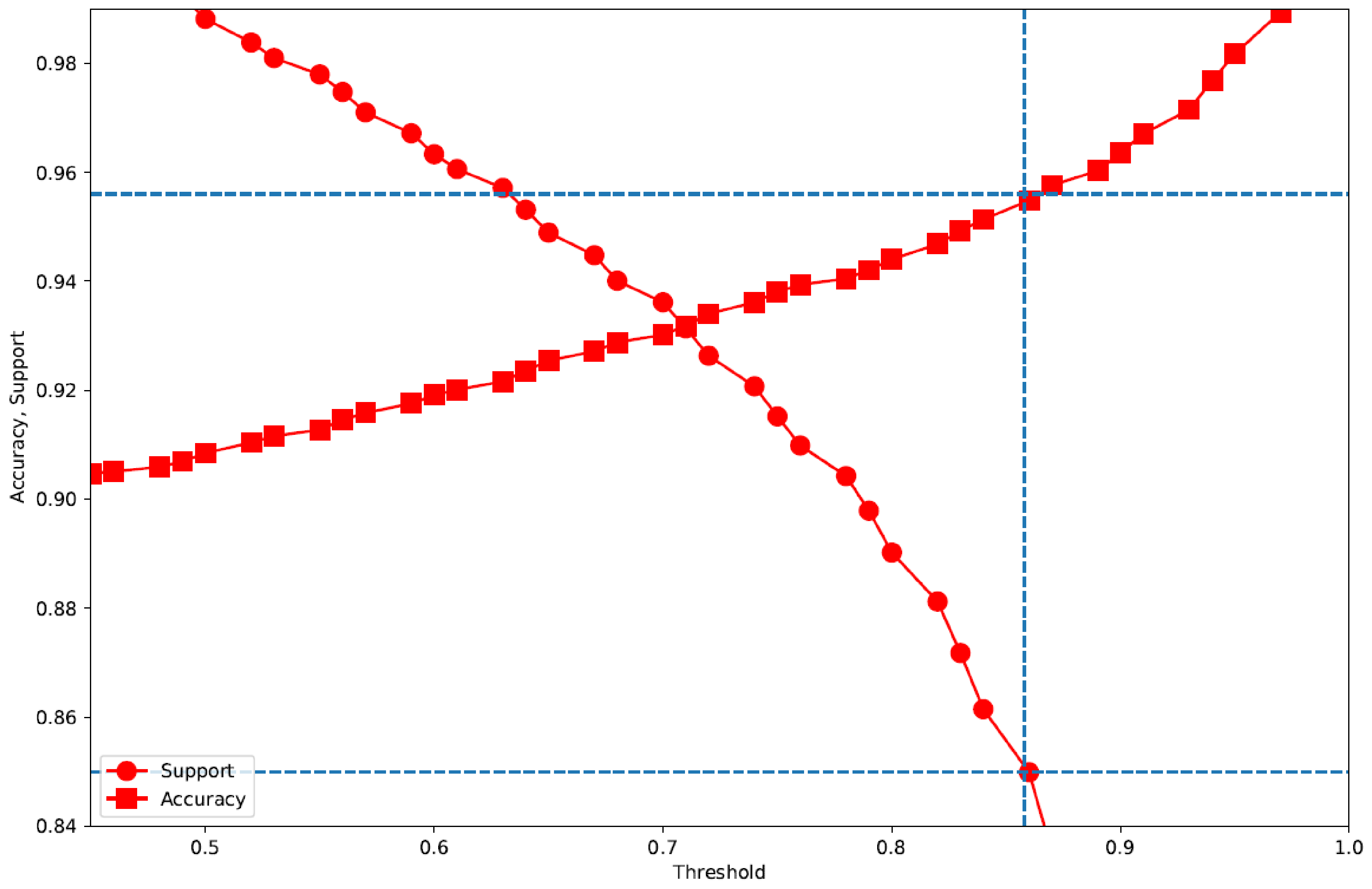
```
Layer (type)                Output Shape          Param #
=================================================================
meta_input (InputLayer)      (None, 15)              0
_________________________________________________________________
dense_8 (Dense)             (None, 20)             320
_________________________________________________________________
dropout_8 (Dropout)          (None, 20)             0
_________________________________________________________________
main_output (Dense)          (None, 5)              105
=================================================================
Total params: 425
```

Ensemble accuracy on 100%

```
Accuracy total model:
 0.9033180469382249
Final Classification Report:
             precision    recall  f1-score   support

         SI       0.92      0.90      0.91      3397
         RE        0.97      0.98      0.98      2299
          D       0.89      0.90      0.90      7143
          C       0.88      0.90      0.89      5578
         AB        0.00      0.00      0.00       118

avg / total       0.90      0.90      0.90     18535
```

Ensemble accuracy on 85%

```
Accuracy total model:
 0.9547993905535805
Final Classification Report:
              precision    recall  f1-score   support

 SI           0.97       0.96       0.96       2942
 RE             0.98       0.99       0.99       2226
 D            0.95       0.96       0.95       5930
 C            0.94       0.95       0.95       4576
 AB             0.00       0.00       0.00         78

avg / total       0.95       0.95       0.95     15752
```

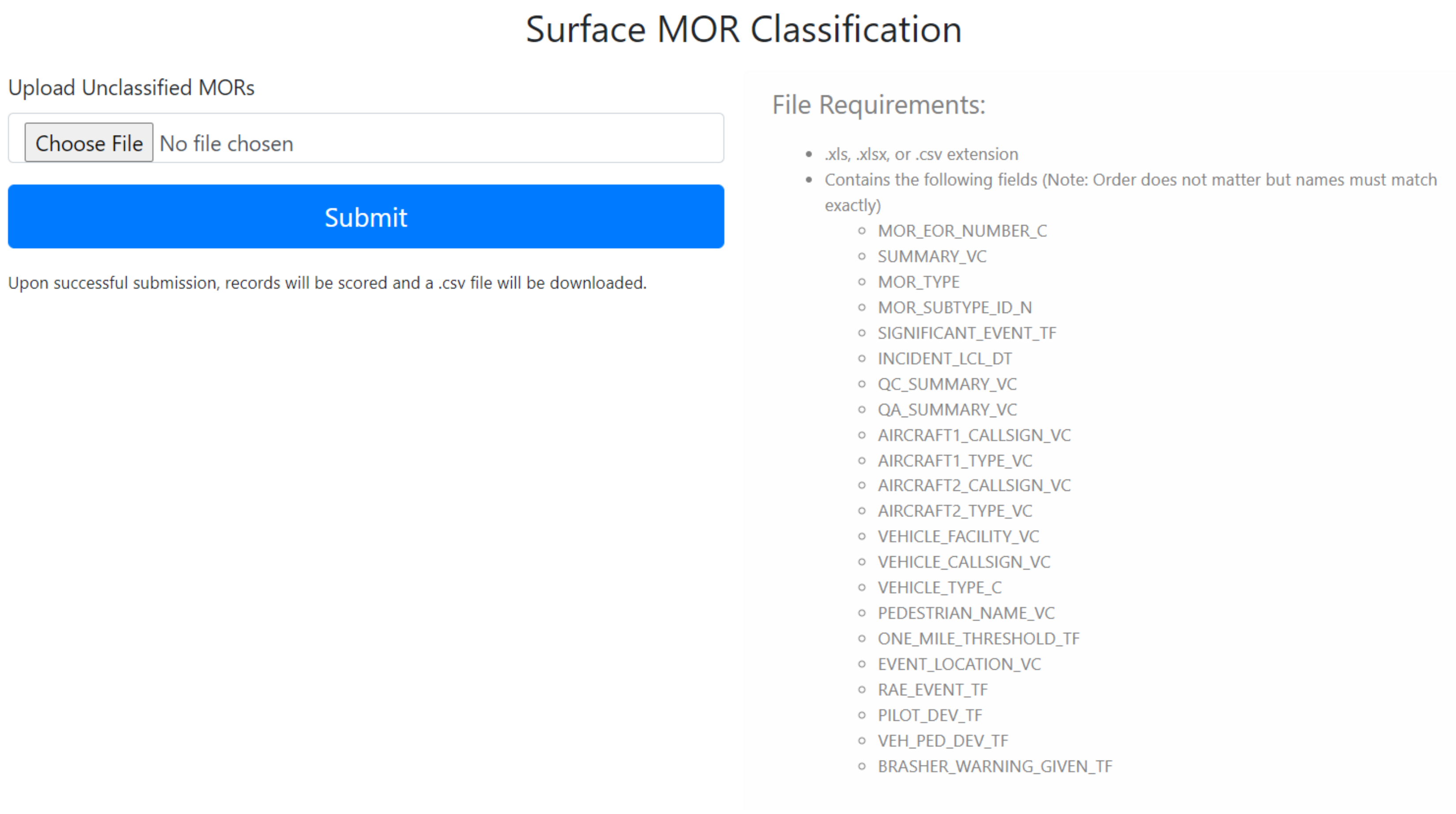


Accuracy, Support
Threshold
0.84
0.86
0.88
0.90
0.92
0.94
0.96
0.98
0.5
0.6
0.7
0.8
0.9
1.0
Support
Accuracy

Where to find

- **Web page:** https://apps.eim.faa.gov/smc/
- **REST API:** https://apps.eim.faa.gov/smc/score_json

Web interface



Surface MOR Classification

Upload Unclassified MORs

Upon successful submission, records will be scored and a .csv file will be downloaded.

File Requirements:

- .xls, .xlsx, or .csv extension
- Contains the following fields (Note: Order does not matter but names must match exactly)
  - MOR_EOR_NUMBER_C
  - SUMMARY_VC
  - MOR_TYPE
  - MOR_SUBTYPE_ID_N
  - SIGNIFICANT_EVENT_TF
  - INCIDENT_LCL_DT
  - QC_SUMMARY_VC
  - QA_SUMMARY_VC
  - AIRCRAFT1_CALLSIGN_VC
  - AIRCRAFT1_TYPE_VC
  - AIRCRAFT2_CALLSIGN_VC
  - AIRCRAFT2_TYPE_VC
  - VEHICLE_FACILITY_VC
  - VEHICLE_CALLSIGN_VC
  - VEHICLE_TYPE_C
  - PEDESTRIAN_NAME_VC
  - ONE_MILE_THRESHOLD_TF
  - EVENT_LOCATION_VC
  - RAE_EVENT_TF
  - PILOT_DEV_TF
  - VEH_PED_DEV_TF
  - BRASHER_WARNING_GIVEN_TF

Source code

- **Model:** **https://git.faa.gov/scm/smc/smc_classifier.git**
- **API: https://git.faa.gov/scm/sma/smc_api.git**

Acknowledgements

- Firdu Bati
- Ramesh Adhikari
- Stephen Roberts
- Yunsun Li
- Mohanad Barhoum
- Nicholas Scialli

[AJI333A-93] add logging for exceptions thrown in SMC wsgi code

Created: 05/18/2023 3:01 PM - Updated: 07/19/2023 12:48 PM - Resolved: 07/19/2023 12:48 PM

Status:	Done
Project:	AJI-333 Analytics
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Improvement	**Priority:**	Medium
Reporter:	Filippov, Igor V (FAA)	**Assignee:**	Filippov, Igor V (FAA)
Resolution:	Implemented	**Votes:**	0
Labels:	None		
Original Estimate:	Not Specified		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		

Field Tab	
Epic Link:	SMC

Description

On rare occasions SMC inference web service seem to fail. It would be good to have more information about the causes of the failure.

Comments

Filippov, Igor V (FAA) added a comment - 05/23/2023 7:50 AM

Adhikari, Ramesh CTR (FAA) I created a pull request, please take a loook.

Filippov, Igor V (FAA) added a comment - 05/23/2023 10:21 AM

Pull request merged.

Tag v4.0 created.

Filippov, Igor V (FAA) added a comment - 06/20/2023 8:41 AM

From Amodeo, John CTR (FAA) :

Updated code baseline has been deployed to PROD: https://ansible.faa.gov/#/jobs/playbook/1173371/output

Please note – after deployment, SMC application was returning a 500 error. I was able to troubleshoot and resolve based on the errors found in /var/log/httpd/error_log

I had to manually create the following file and set “chmod 666” on the file so the webserver could write to it:

/var/www/wsgi/smor_api.log

After performing this step, everything appears to be working.

Please update the automation next time you are in there.

Adhikari, Ramesh CTR (FAA) added a comment - 07/19/2023 9:43 AM

Ansible script updated for /var/www/wsgi/smor_api.log file with mode 666.

Filippov, Igor V (FAA) added a comment - 07/19/2023 12:48 PM

the logging implemented and deployed.

[AJI333A-33] Re-train autoclass models with training datasets (add last three years data)

Created: 03/23/2021 10:40 AM - Updated: 07/22/2021 3:44 PM - Resolved: 07/22/2021 3:44 PM

Status:	Done
Project:	AJI-333 Analytics
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Improvement	**Priority:**	Medium
Reporter:	Adhikari, Ramesh CTR (FAA)	**Assignee:**	Adhikari, Ramesh CTR (FAA)
Resolution:	Fixed	**Votes:**	0
Labels:	None		
Original Estimate:	Not Specified		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		

Field Tab	
Epic Link:	SMC

Description

Existing models are trained data from 2012 to Jan/2018. Since, there are data available for last three years, we will re-train the models with additional data.

Comments

Adhikari, Ramesh CTR (FAA) added a comment - 04/07/2021 9:36 AM

Bati, Firdu (FAA), [~Igor V-CTR Filippov]

- Tuned the parameters for different models (svm, lgb, lstm, cnn, cnn-lstm, and cnn_bilstm) using additional training data. Codes and output logs are added in the repo as re_train folder.
- Individual model performance are similar to the existing models, however, the 'manual review' items seem to be limited with the retrained models.

Adhikari, Ramesh CTR (FAA) added a comment - 06/28/2021 9:23 AM

Several machine learning and deep learning models were trained with additional data sets. The trained models, cross-validation reports and parameter tunings have been uploaded in the git repo 'new_updates' branch.

Adhikari, Ramesh CTR (FAA) added a comment - 07/22/2021 3:40 PM

Re-training of the models with additional data were completed. Performance of the models are saved as accuracy and confusion matrix for random sample cross-validation and yearly sample cross validation. Code files, saved trained models and other data files are saved and updated in the repository. Class probabilities from every models for all available events are saved/uploaded in repository as csv files.

Adhikari, Ramesh CTR (FAA) added a comment - 07/22/2021 3:44 PM

Updated files/data after training ML/DL models additional available training data are uploaded in the git repository. Saved upgraded models were used by SMC_API.

[AJI333A-32] Autoclass Refinements

Created: 03/22/2021 2:25 PM - Updated: 08/11/2021 11:57 AM - Resolved: 08/11/2021 11:57 AM

Status:	Done
Project:	AJI-333 Analytics
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Improvement	**Priority:**	Medium
Reporter:	Adhikari, Ramesh CTR (FAA)	**Assignee:**	Adhikari, Ramesh CTR (FAA)
Resolution:	Completed	**Votes:**	0
Labels:	None		
Original Estimate:	Not Specified		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		

Field Tab	
Epic Link:	SMC

Description

Refinements required: 1) data preprocessing, 2) model selections, 3) model training with available additional data, and 4) unification of model outputs.

Comments

Adhikari, Ramesh CTR (FAA) added a comment - 03/22/2021 3:06 PM

1) Added two fields 'AIRCRAFT1_CALLSIGN_VC' and 'AIRCRAFT1_TYPE_VC' in the required_fields.py and changed scoring.py to include both aircrafts information.
2) Also added list of null-like words 'None', 'nan', 'n/a', as possible entity used in the database to indicate the absence of the aircrafts.
3) A sample input data 'Sample_for_scoring_ac1-ac2.csv' also added in the misc folder.

Adhikari, Ramesh CTR (FAA) added a comment - 06/25/2021 3:55 PM

SMC is now updated with the newly trained models.

1) Several models (SVM, LGB, RF, MNB, XGB, CAT, CNN, LSTM, CNNLSTM, BiLSTM, CNNBiLSTM) were trained with the new training data (included data from FY2012 to FY2020).
2) We analyzed the performance of each model on the basis of accuracy, confusion matrix, and classification reports(precision, recall etc.). LGB, XGB, SVM, CAT, CNN were best performing models upon five-fold stratified sample cross validation for each.
3) After extensive analysis of the final output of testing (a) yearly sampled test data and (b) stratified random samples of corresponding size, we found that combination of LGB, SVM and CNN models outperforms the other possible combinations. The unified output were filtered using certain thresholds for confidence level (probability outputs of models). Previous model uses the voting algorithm with set of thresholds depending on predicted classes.
4) We have replaced the manual selection of set of thresholds for voting algorithm by feeding the class probabilities from different models to a simple neural network with one hidden layer. The optimum value of a single threshold for predicting probability was chosen to maintain accuracy ~0.96 (same as average accuracy of existing model).
5) The newly trained models with NN-meta-model for unified output, perform with accuracy ~96% and covers ~80% of the total cases. This is about 10% more coverage than existing models ~70% for similar accuracy.
6) New models and updates have been committed to the git repo and are linked here.

Adhikari, Ramesh CTR (FAA) added a comment - 06/28/2021 9:13 AM

Modification has been uploaded in the git repo 'updates'.

Adhikari, Ramesh CTR (FAA) added a comment - 08/11/2021 11:57 AM

Upgraded Autoclass (version v3.0 in git repo) is in EIM PROD.

[AJI333A-27] Surface MOR Classification

Created: 02/23/2021 11:46 AM - Updated: 09/27/2023 10:54 AM - Resolved: 09/03/2021 11:18 AM

Status:	Done
Project:	AJI-333 Analytics
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Epic	**Priority:**	Medium
Reporter:	Adhikari, Ramesh CTR (FAA)	**Assignee:**	Adhikari, Ramesh CTR (FAA)
Resolution:	Completed	**Votes:**	0
Labels:	None		
Original Estimate:	Not Specified		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		

Field Tab	
Epic Name:	SMC
Epic Link:	

Comments

Filippov, Igor V (FAA) added a comment - 09/03/2021 9:46 AM

Since it has been released and in production now perhaps it makes sense to retire this ticket?

Next release epic could be called SMC 2.0 for example.

Adhikari, Ramesh CTR (FAA) added a comment - 09/03/2021 11:18 AM

This epic is completed. Current version of SMC is running in production now.

[AJI333A-24] Update Commercial/Non-commercial

Created: 01/08/2021 11:13 AM - Updated: 03/22/2021 3:35 PM - Resolved: 03/22/2021 3:35 PM

Status:	Done
Project:	AJI-333 Analytics
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Improvement	**Priority:**	Medium
Reporter:	Adhikari, Ramesh CTR (FAA)	**Assignee:**	Adhikari, Ramesh CTR (FAA)
Resolution:	Fixed	**Votes:**	0
Labels:	None		
Original Estimate:	Not Specified		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		

Field Tab	
Epic Link:	SMC

Description

To classify the events associated with commercial or non-commercial.

Links

Familial

Is Child of	[~~AJI333A-1~~]	Deployment of SMC in EIM PROD	Done

Comments

Filippov, Igor V (FAA) added a comment - 01/12/2021 3:08 PM

Adhikari, Ramesh CTR (FAA)

I looked at the code, good work.

A few suggestions, if I may:

1. I don't like the additional config file smor_new.ini. If we do it with each new update are we going to have smor_new_new_new.ini soon or is it going to be smor_new2.ini or something else? There is no need to reinvent versioning system with filenames.
2. You changed the output in test_console.py to 'OUtput_with_commercial_flag.csv'. I am not sure I like the hard-coded file name in the output, especially with a strange capitalization schema. If you don't like output to stdout then add a command line parameter for the output file.
3. You are merging in the result of get_commercial_from_callsign in scoring.py just before the call to set_tshold, and then in set_tshold you are merging it again. Wouldn't it make more sense to merge it just once, in scoring.py **after** set_tshold? This way voting_score.py would not be modified at all - and it shouldn't be.
4. In fact, I would propose to get rid of merging at all. Compute commercial/non-commercial in the beginning, before the preprocessing, then just carry those columns throughout. But this is optional, if it's too much trouble just compute them in a separate dataframe (before the preprocessing), then merge it in after set_tshold.

Adhikari, Ramesh CTR (FAA) added a comment - 01/13/2021 11:27 AM

https://jira.faa.gov/secure/ViewProfile.jspa?name=Igor+V-CTR+Filippov
Thank you so much Igor for fruitful insights. Your suggestion is followed as:

1) changed smor_new.ini to smor.ini
2) removed the hard-coded filename
3) or 4) called get_commercial_from_callsign in scoring.py just after set_tshold as suggested in comment 4.

Filippov, Igor V (FAA) added a comment - 01/13/2021 12:29 PM

Adhikari, Ramesh CTR (FAA)

This looks much better. A couple more things to make it production ready. The 2nd problem was actually in the original code, but we might as well address it now.

1. Please add a new sample input file in /data folder with all fields present and modify prompt in test_console.py accordingly.
2. Right now if not all required fields present preprocess() function returns False in line 39, where the scoring function expects DataFrame. I think it would be more clear if instead of returning a wrong datatype an exception would be thrown. You can use raise() or assert(): https://realpython.com/python-exceptions/

I also think I found an old bug on line 244, but it looks like the output of this function is not used anywhere, so I'm not sure why it is there in the first place.

Adhikari, Ramesh CTR (FAA) added a comment - 01/14/2021 2:50 PM

igor.v-ctr.filippov@faa.gov
1) Input samples are added in /scoring/data/sample_inputs folder.
2) 'return False' is replaced by raise Exception ("......")

Filippov, Igor V (FAA) added a comment - 01/14/2021 3:07 PM

Looks great now!

Good work!

Adhikari, Ramesh CTR (FAA) added a comment - 01/14/2021 3:12 PM

The commercial/non-commercial aircrafts involved in the events are recorded based on aircraft callsign and type. We have used the same function and key-files used in ASM.

Adhikari, Ramesh CTR (FAA) added a comment - 03/22/2021 3:35 PM

Commercial/non-commercial flags were added and merged to the master branch. Note: the smc_api in production is running without commercial/non-commercial flag and saved as v2.0 in the git repo.

[AJI333A-1] Deployment of SMC in EIM PROD

Created: 10/13/2020 6:33 PM - Updated: 02/23/2021 11:52 AM - Resolved: 02/05/2021 12:51 PM

Status:	Done
Project:	AJI-333 Analytics
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	New Feature	**Priority:**	High
Reporter:	Adhikari, Ramesh CTR (FAA)	**Assignee:**	Adhikari, Ramesh CTR (FAA)
Resolution:	Fixed	**Votes:**	0
Labels:	None		
Original Estimate:	32h		
Remaining Estimate:	8h		
Time Spent:	Not Specified		

Field Tab	
Epic Link:	SMC

Description

Deploy the Surface MOR project using ansible. Apply ssl certificate for https secure network connection.

Links

Familial

Is Parent of	[~~AJI333A-24~~]	Update Commercial/Non-commercial	Done

Comments

Adhikari, Ramesh CTR (FAA) added a comment - 10/15/2020 9:27 AM

This project is passed to test in EIM

Adhikari, Ramesh CTR (FAA) added a comment - 02/05/2021 12:51 PM

Version v2.0 is now running in EIM PROD.

Surface MOR Auto Classification

Federal Aviation Administration
Federal Aviation Administration

Presented to:	ML WG
By:	Firdu Bati, PhD
Date:	June 2019

AJI 333 – Performance & Analytics

- Provide analytical support to safety & technical training
- Comprised of Data Scientists, Mathematicians, OR Analysts
- Focus on quantitative risk estimate, machine learning, mathematical modeling & simulation.

ML - Areas of Deployment

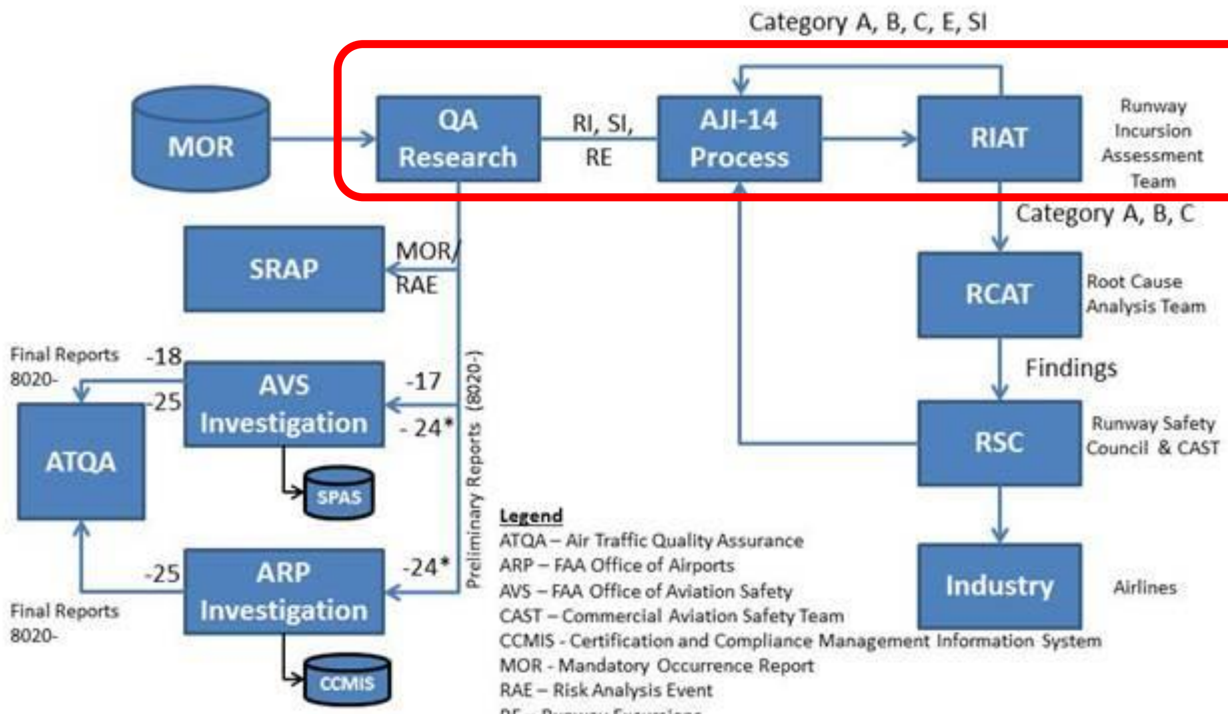
- **Event detection for Surface metric**
  - RE, RI, SI
- **Event detection for airborne metric**
  - CFIT, UCFIT, Turbulence
- **RAP Factors & Barriers relationships**
  - Apriori to extract rules
- **Surface MOR severity classification**
  - Cat A, B, C & D, SI, RE

Overview – SMOR Classification

- **The surface MOR classification task was borne out of a desire to reduce the amount of manual event review**
- **We used knowledge gained when deploying the SSM**
- **SMEs and analysts in the Runway Safety Office currently manually review approximately 2,500 MORs per year to classify RI, SI, RE events**

Current Surface MOR Classification

Surface Event Assessment Flow Chart (Enhanced View)



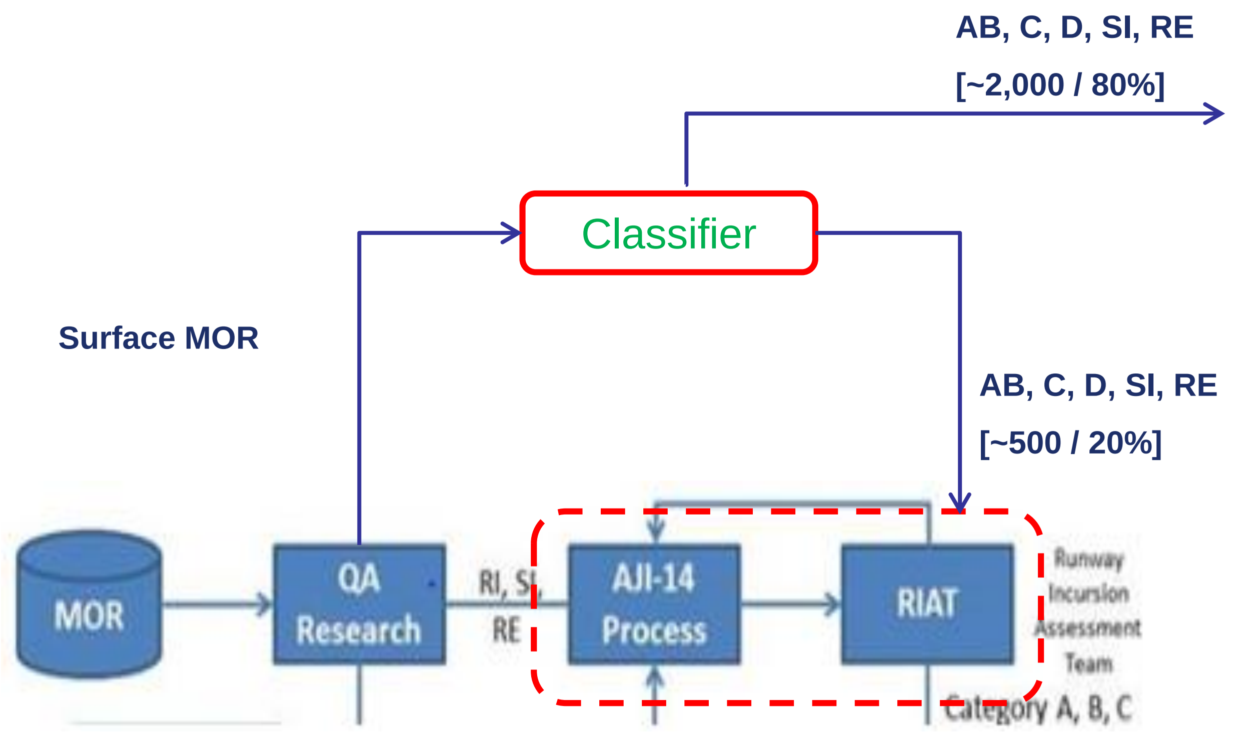
Legend

- ATQA – Air Traffic Quality Assurance
- ARP – FAA Office of Airports
- AVS – FAA Office of Aviation Safety
- CAST – Commercial Aviation Safety Team
- CCMIS - Certification and Compliance Management Information System
- MOR - Mandatory Occurrence Report
- RAE – Risk Analysis Event
- RE – Runway Excursions
- RI – Runway Incursions
- SI – Surface Incidents
- SPAS – Safety Performance Analysis System
- SRAP – Surface Risk Analysis Process

- Mechanic Taxi events reported on the -24 and investigated by AVS

** Pilot Surface events reported on the -17

Enhanced Surface MOR Classification



Modeling Approach

- **Feature representation: tf_idf, word embedding (word2vec)**
- **Three Machine Learning Models:**
  - Gradient Boosting Machine (GBM)
  - Support Vector Machine (SVM)
  - Deep Learning – (CNN/RNN/LSTM)
- **Applied Merging Logic:**
  - Three Matches: lower average conf. threshold (0.75)
  - Two matches : medium average conf. threshold (0.85)
  - No match: higher conf. threshold (0.95)

Cleaning/Preprocessing

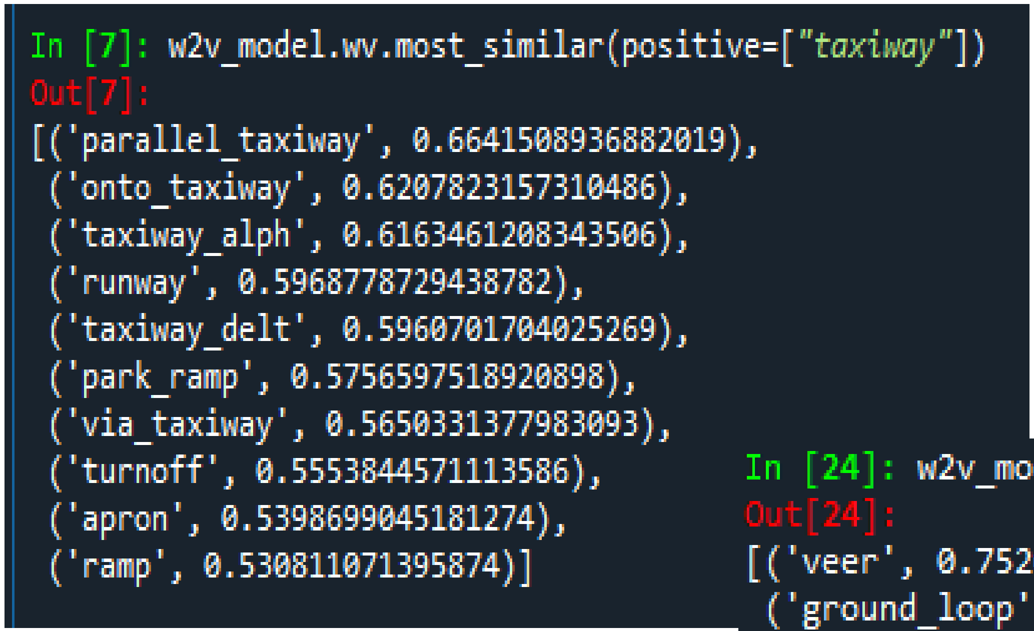
- **Remove punctuation, remove numbers, strip white spaces, change to lower case, remove stop words**
- **Dictionary: developed a dictionary to unify terms.**

Term	Replacement
(air|aero) ?(craft|plane)s?	airplane
(runway|rnway|rnwy)	runway
(ran of|rolled off|rolled of)	ranoff
(tkof|tk off|take(s)? off|taking off|took off)	takeoff

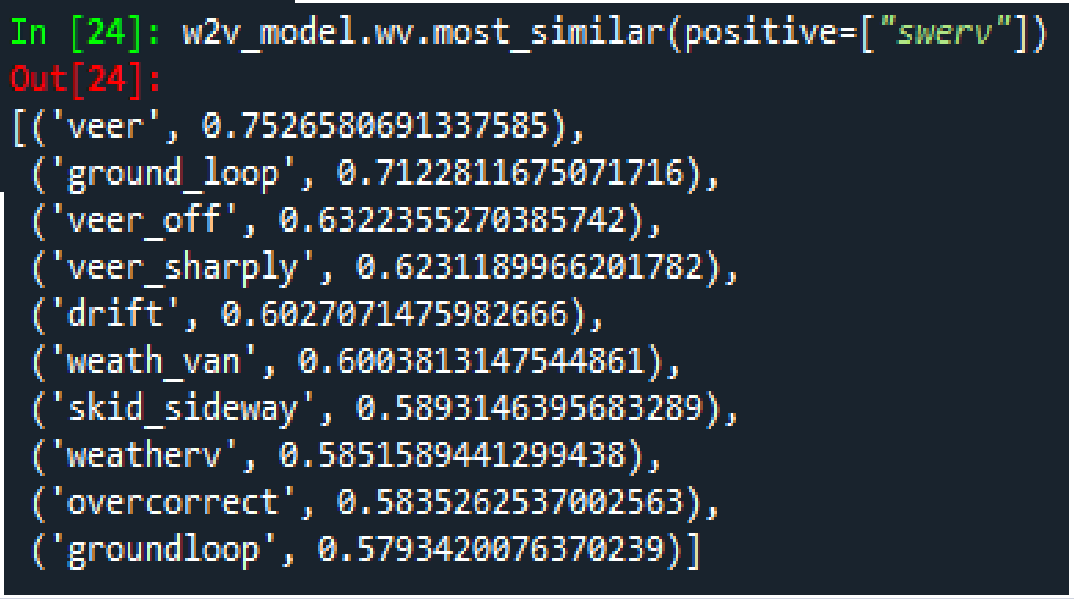
Features from Text

- TF-IDF weighting: numerical statistic to capture the importance of a word/term to a document in a corpus
- Word2vec: a type of word embeddings which produces a vector for every word
- Combining the two can be done by averaging a word vector and weighting by a TF-IDF term

Terms Similarity



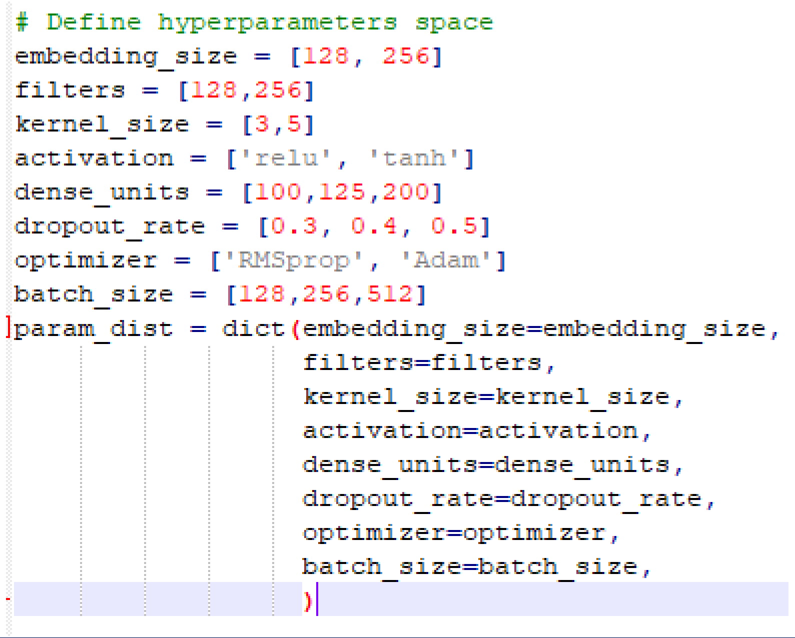
```
In [7]: w2v_model.wv.most_similar(positive=["taxiway"])
Out[7]:
[('parallel_taxiway', 0.6641508936882019),
 ('onto_taxiway', 0.620782315731048),
 ('taxiway_alph', 0.6163461208343506),
 ('runway', 0.5968778729438782),
 ('taxiway_delt', 0.5960701704025269),
 ('park_ramp', 0.5756597518920898),
 ('via_taxiway', 0.5650331377983093),
 ('turnoff', 0.5553844571113586),
 ('apron', 0.5398699045181274),
 ('ramp', 0.530811071395874)]
```



```
In [24]: w2v_model.wv.most_similar(positive=["swerv"])
Out[24]:
[('veer', 0.7526580691337585),
 ('ground_loop', 0.7122811675071716),
 ('veer_off', 0.6322355270385742),
 ('veer_sharply', 0.6231189966201782),
 ('drift', 0.6027071475982666),
 ('weath_van', 0.6003813147544861),
 ('skid_sideway', 0.5893146395683289),
 ('weatherv', 0.5851589441299438),
 ('overcorrect', 0.5835262537002563),
 ('groundloop', 0.5793420076370239)]
```

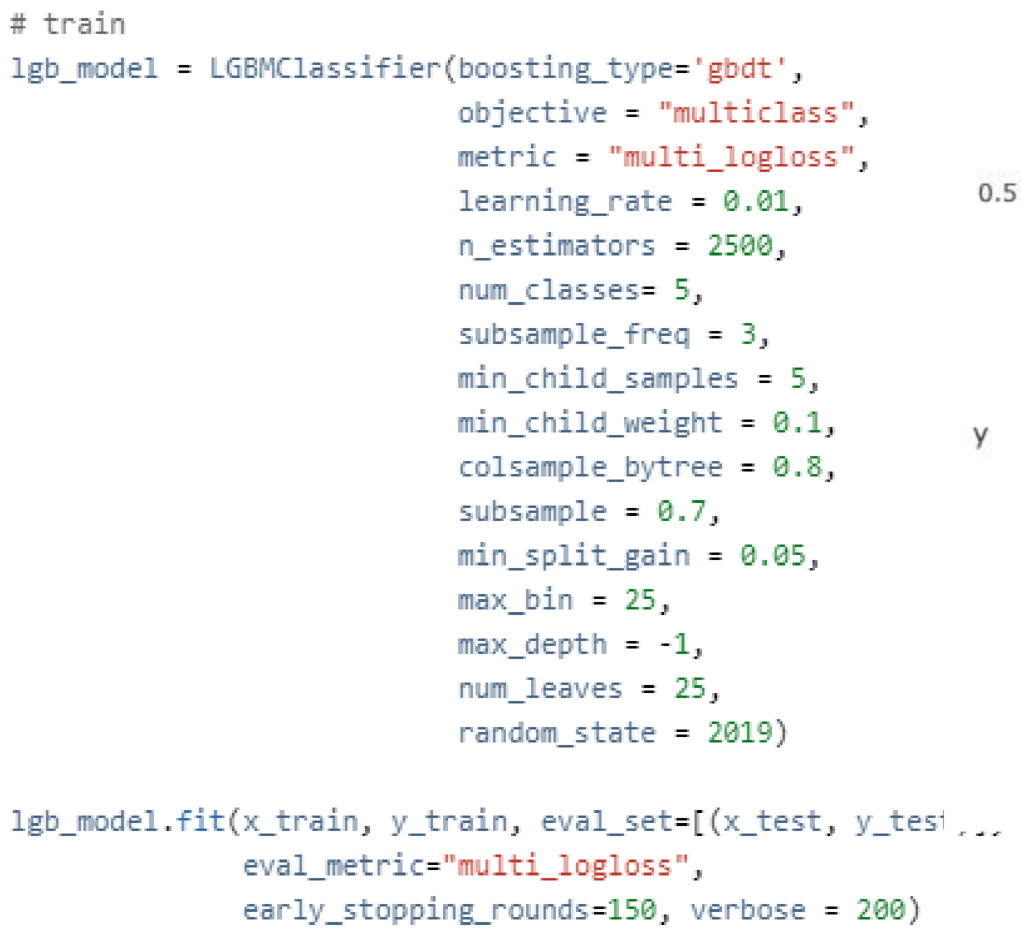
Hyper-Parameters Search

- **Performed an extensive parameters search to tune each model independently.**
- **Independent test set to evaluate models.**
  - 5-fold cross-validation

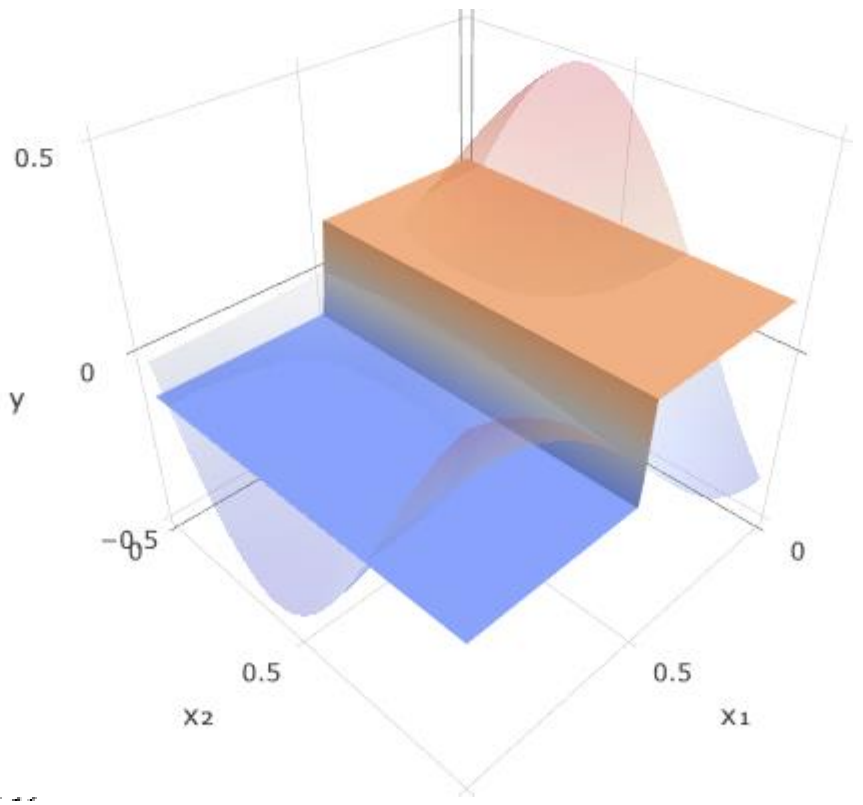


```
# Define hyperparameters space
embedding_size = [128, 256]
filters = [128,256]
kernel_size = [3,5]
activation = ['relu', 'tanh']
dense_units = [100,125,200]
dropout_rate = [0.3, 0.4, 0.5]
optimizer = ['RMSprop', 'Adam']
batch_size = [128,256,512]
param_dist = dict(embedding_size=embedding_size,
                  filters=filters,
                  kernel_size=kernel_size,
                  activation=activation,
                  dense_units=dense_units,
                  dropout_rate=dropout_rate,
                  optimizer=optimizer,
                  batch_size=batch_size,
                  )
```

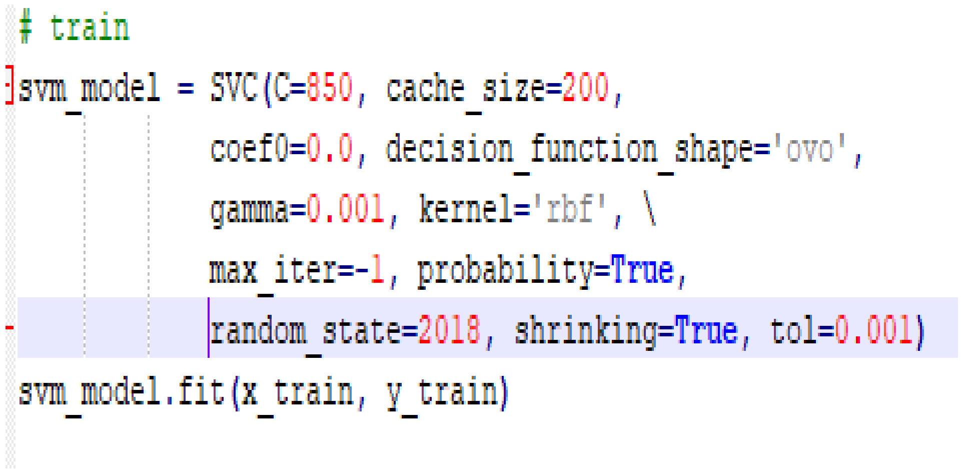
GBM



```
# train
lgb_model = LGBMClassifier(boosting_type='gbdt',
                           objective = "multiclass",
                           metric = "multi_logloss",
                           learning_rate = 0.01,
                           n_estimators = 2500,
                           num_classes= 5,
                           subsample_freq = 3,
                           min_child_samples = 5,
                           min_child_weight = 0.1,
                           colsample_bytree = 0.8,
                           subsample = 0.7,
                           min_split_gain = 0.05,
                           max_bin = 25,
                           max_depth = -1,
                           num_leaves = 25,
                           random_state = 2019)

lgb_model.fit(x_train, y_train, eval_set=[(x_test, y_test ...
              eval_metric="multi_logloss",
              early_stopping_rounds=150, verbose = 200)
```

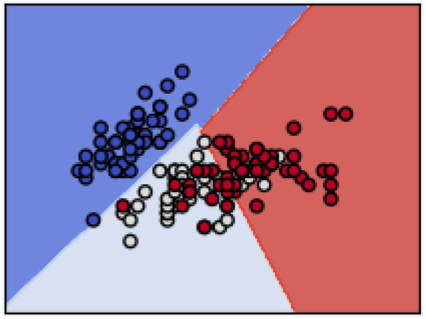


SVM

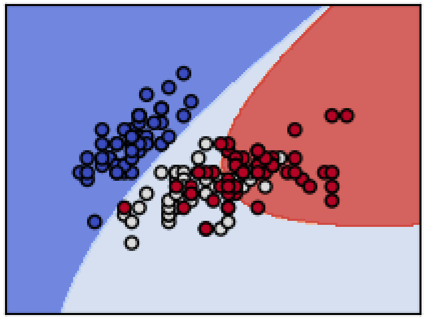


```
# train
svm_model = SVC(C=850, cache_size=200,
                coef0=0.0, decision_function_shape='ovo',
                gamma=0.001, kernel='rbf', \
                max_iter=-1, probability=True,
                random_state=2018, shrinking=True, tol=0.001)
svm_model.fit(x_train, y_train)
```

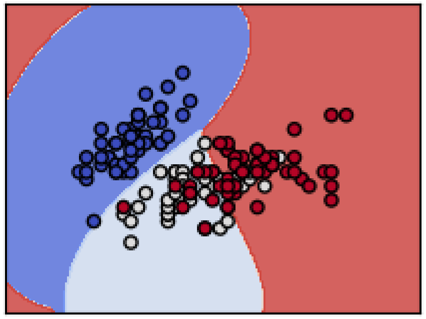
Linear Kernel



RBF Kernel



Polynomial Kernel



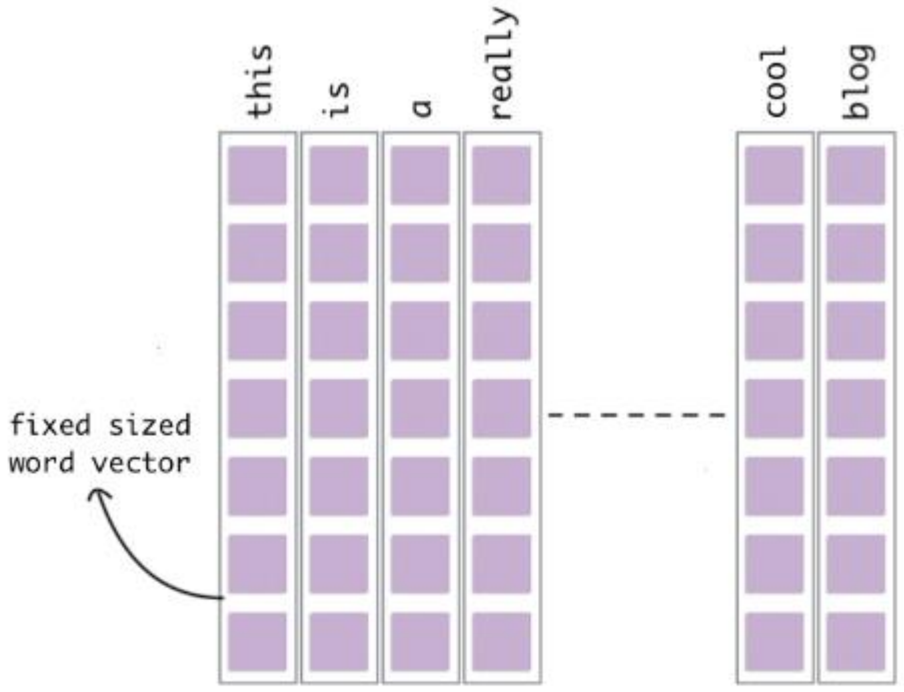
Federal Aviation Administration

CNN/RNN/LSTM

```
nlp_input = Input(shape=(100,), name='nlp_input')
emb = Embedding(input_dim=5000, output_dim=64, input_length=100)(nlp_input)
nlp_out = Conv1D(filters,
                 kernel_size,
                 padding='valid',
                 activation='relu',
                 strides=1)(emb)
```

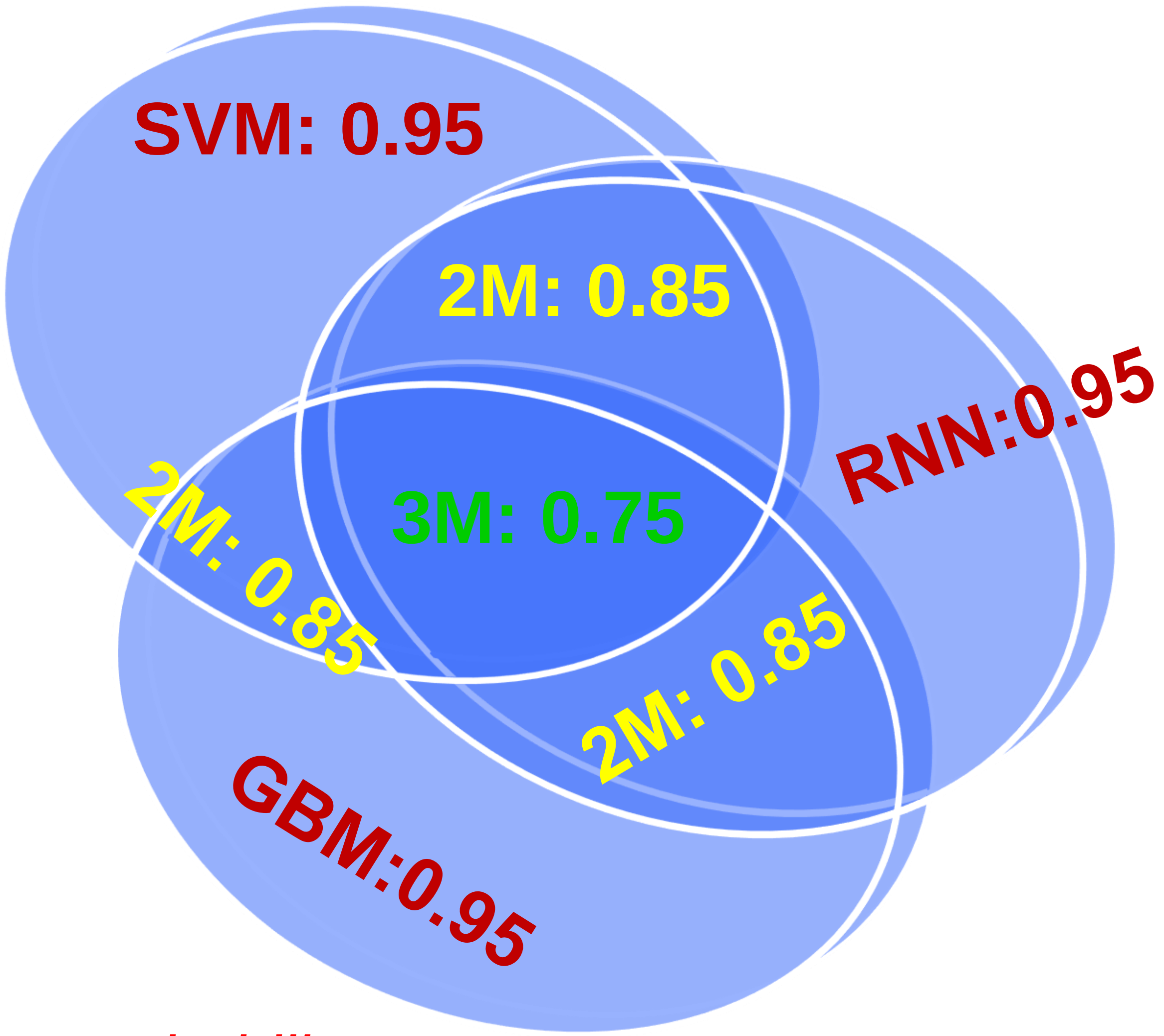
```
x = keras.layers.concatenate([nlp_out, meta_input])
x = Dense(250, activation='relu')(x)
x = Dropout(0.3)(x)
x = Dense(200, activation='relu')(x)
x = Dropout(0.4)(x)
x = Dense(250, activation='relu')(x)
x = Dropout(0.5)(x)
```

```
main_output = Dense(5, activation='softmax', name='main_output')
model = Model(inputs=[nlp_input, meta_input], output=main_output
model.compile(optimizer='adam',
              metrics=['accuracy'],
              loss={'main_output': 'categorical_crossentropy'})
```



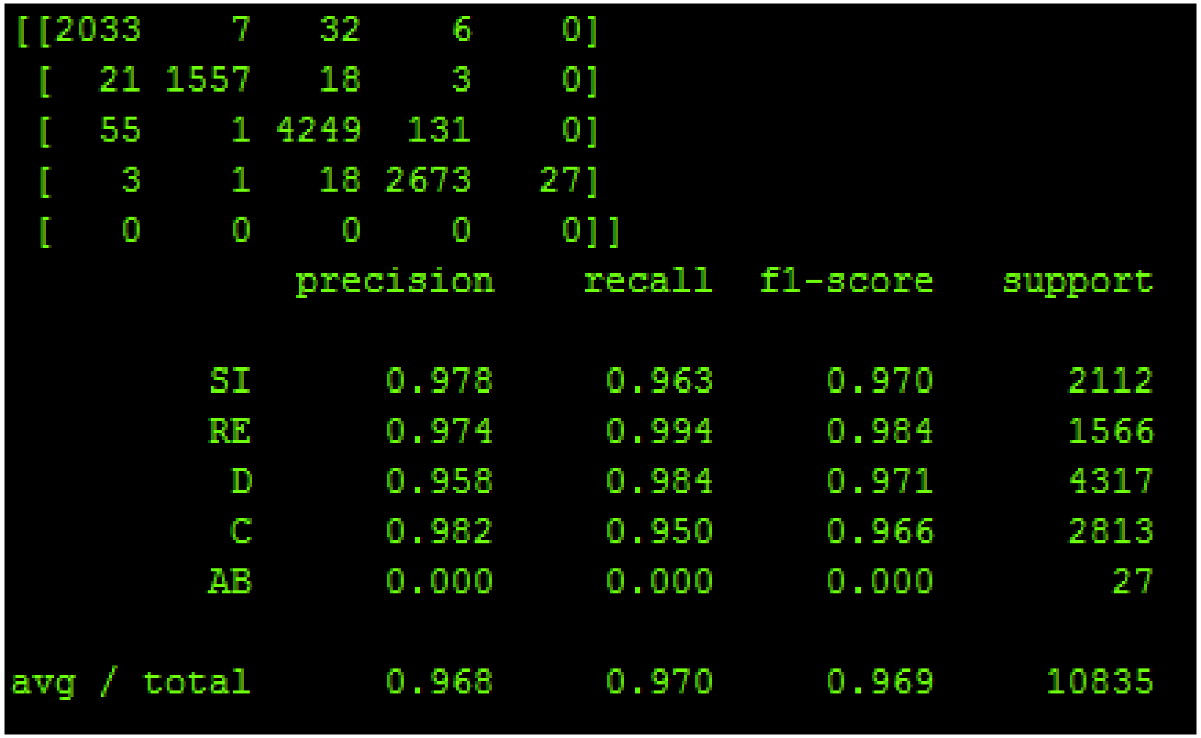
Text data as Time Series

Classification Threshold



Average prediction probability

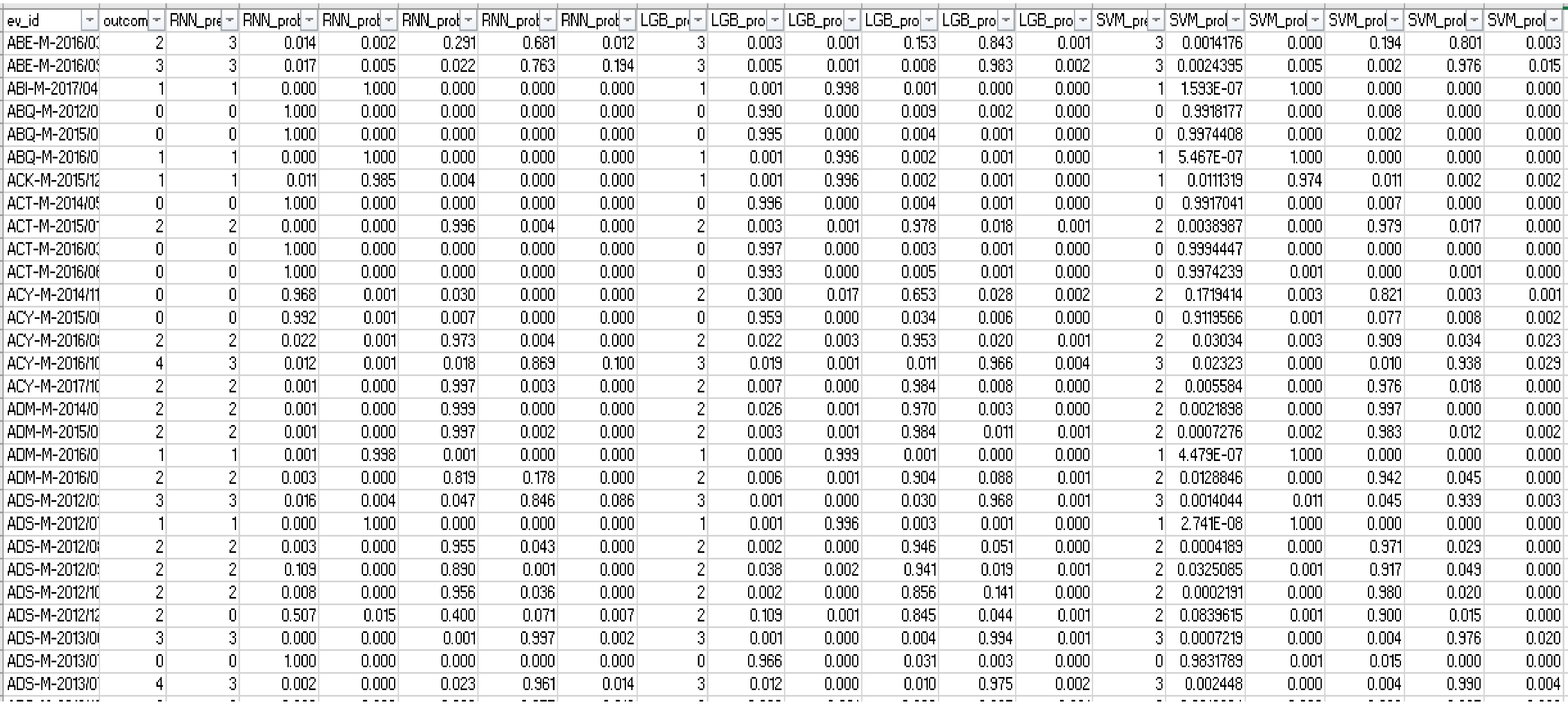
Model Performance



```
[[2033    7   32    6    0]
 [  21 1557   18    3    0]
 [  55    1 4249  131    0]
 [   3    1   18 2673   27]
 [   0    0    0    0    0]]
             precision    recall  f1-score   support

         SI      0.978     0.963     0.970      2112
         RE      0.974     0.994     0.984      1566
          D      0.958     0.984     0.971      4317
          C      0.982     0.950     0.966      2813
         AB      0.000     0.000     0.000        27

avg / total      0.968     0.970     0.969     10835
```

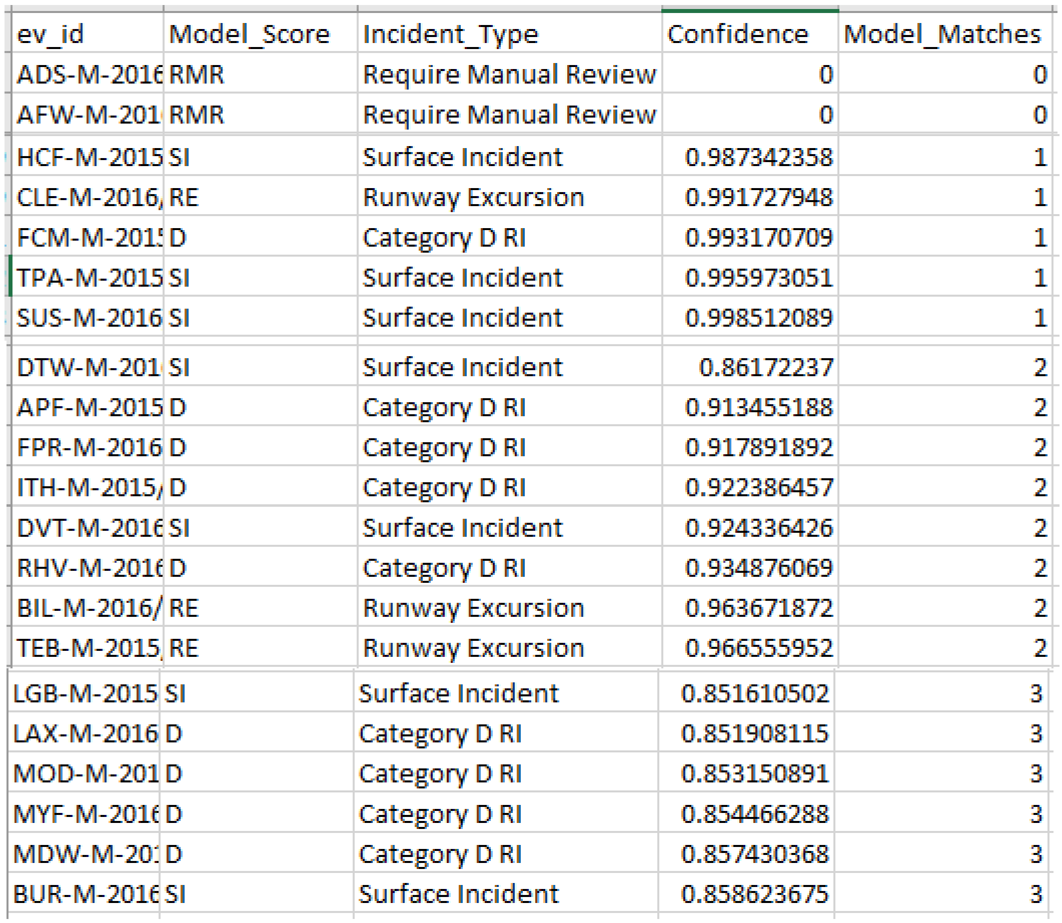
Using threshold: Accuracy 82% to 97%,
Coverage 100% to 80% coverage

Individual Model Prediction



ev_id	outcom	RNN_pre	RNN_prob	RNN_prob	RNN_prob	RNN_prob	RNN_prob	LGB_pr	LGB_pro	LGB_pro	LGB_pro	LGB_pro	LGB_pro	SVM_pre	SVM_prob	SVM_prob	SVM_prob	SVM_prob	SVM_prob
ABE-M-2016/0	2	3	0.014	0.002	0.291	0.681	0.012	3	0.003	0.001	0.153	0.843	0.001	3	0.0014176	0.000	0.194	0.801	0.003
ABE-M-2016/0	3	3	0.017	0.005	0.022	0.763	0.194	3	0.005	0.001	0.008	0.983	0.002	3	0.0024395	0.005	0.002	0.976	0.015
ABI-M-2017/04	1	1	0.000	1.000	0.000	0.000	0.000	1	0.001	0.998	0.001	0.000	0.000	1	1.593E-07	1.000	0.000	0.000	0.000
ABQ-M-2012/0	0	0	1.000	0.000	0.000	0.000	0.000	0	0.990	0.000	0.009	0.002	0.000	0	0.9918177	0.000	0.008	0.000	0.000
ABQ-M-2015/0	0	0	1.000	0.000	0.000	0.000	0.000	0	0.995	0.000	0.004	0.001	0.000	0	0.9974408	0.000	0.002	0.000	0.000
ABQ-M-2016/0	1	1	0.000	1.000	0.000	0.000	0.000	1	0.001	0.996	0.002	0.001	0.000	1	5.467E-07	1.000	0.000	0.000	0.000
ACK-M-2015/1	1	1	0.011	0.985	0.004	0.000	0.000	1	0.001	0.996	0.002	0.001	0.000	1	0.0111319	0.974	0.011	0.002	0.002
ACT-M-2014/0	0	0	1.000	0.000	0.000	0.000	0.000	0	0.996	0.000	0.004	0.001	0.000	0	0.9917041	0.000	0.007	0.000	0.000
ACT-M-2015/0	2	2	0.000	0.000	0.996	0.004	0.000	2	0.003	0.001	0.978	0.018	0.001	2	0.0038987	0.000	0.979	0.017	0.000
ACT-M-2016/0	0	0	1.000	0.000	0.000	0.000	0.000	0	0.997	0.000	0.003	0.001	0.000	0	0.9994447	0.000	0.000	0.000	0.000
ACT-M-2016/0	0	0	1.000	0.000	0.000	0.000	0.000	0	0.993	0.000	0.005	0.001	0.000	0	0.9974239	0.001	0.000	0.001	0.000
ACY-M-2014/11	0	0	0.968	0.001	0.030	0.000	0.000	2	0.300	0.017	0.653	0.028	0.002	2	0.1719414	0.003	0.821	0.003	0.001
ACY-M-2015/0	0	0	0.992	0.001	0.007	0.000	0.000	0	0.959	0.000	0.034	0.006	0.000	0	0.9119566	0.001	0.077	0.008	0.002
ACY-M-2016/0	2	2	0.022	0.001	0.973	0.004	0.000	2	0.022	0.003	0.953	0.020	0.001	2	0.03034	0.003	0.909	0.034	0.023
ACY-M-2016/1	4	3	0.012	0.001	0.018	0.869	0.100	3	0.019	0.001	0.011	0.966	0.004	3	0.02323	0.000	0.010	0.938	0.029
ACY-M-2017/1	2	2	0.001	0.000	0.997	0.003	0.000	2	0.007	0.000	0.984	0.008	0.000	2	0.005584	0.000	0.976	0.018	0.000
ADM-M-2014/0	2	2	0.001	0.000	0.999	0.000	0.000	2	0.026	0.001	0.970	0.003	0.000	2	0.0021898	0.000	0.997	0.000	0.000
ADM-M-2015/0	2	2	0.001	0.000	0.997	0.002	0.000	2	0.003	0.001	0.984	0.011	0.001	2	0.0007276	0.002	0.983	0.012	0.002
ADM-M-2016/0	1	1	0.001	0.998	0.001	0.000	0.000	1	0.000	0.999	0.001	0.000	0.000	1	4.479E-07	1.000	0.000	0.000	0.000
ADM-M-2016/0	2	2	0.003	0.000	0.819	0.178	0.000	2	0.006	0.001	0.904	0.088	0.001	2	0.0128846	0.000	0.942	0.045	0.000
ADS-M-2012/0	3	3	0.016	0.004	0.047	0.846	0.086	3	0.001	0.000	0.030	0.968	0.001	3	0.0014044	0.011	0.045	0.939	0.003
ADS-M-2012/0	1	1	0.000	1.000	0.000	0.000	0.000	1	0.001	0.996	0.003	0.001	0.000	1	2.741E-08	1.000	0.000	0.000	0.000
ADS-M-2012/0	2	2	0.003	0.000	0.955	0.043	0.000	2	0.002	0.000	0.946	0.051	0.000	2	0.0004189	0.000	0.971	0.029	0.000
ADS-M-2012/0	2	2	0.109	0.000	0.890	0.001	0.000	2	0.038	0.002	0.941	0.019	0.001	2	0.0325085	0.001	0.917	0.049	0.000
ADS-M-2012/1	2	2	0.008	0.000	0.956	0.036	0.000	2	0.002	0.000	0.856	0.141	0.000	2	0.0002191	0.000	0.980	0.020	0.000
ADS-M-2012/1	2	0	0.507	0.015	0.400	0.071	0.007	2	0.109	0.001	0.845	0.044	0.001	2	0.0839615	0.001	0.900	0.015	0.000
ADS-M-2013/0	3	3	0.000	0.000	0.001	0.997	0.002	3	0.001	0.000	0.004	0.994	0.001	3	0.0007219	0.000	0.004	0.976	0.020
ADS-M-2013/0	0	0	1.000	0.000	0.000	0.000	0.000	0	0.966	0.000	0.031	0.003	0.000	0	0.9831789	0.001	0.015	0.000	0.000
ADS-M-2013/0	4	3	0.002	0.000	0.023	0.961	0.014	3	0.012	0.000	0.010	0.975	0.002	3	0.002448	0.000	0.004	0.990	0.004

Final Model Prediction



ev_id	Model_Score	Incident_Type	Confidence	Model_Matches
ADS-M-2016	RMR	Require Manual Review	0	0
AFW-M-201	RMR	Require Manual Review	0	0
HCF-M-2015	SI	Surface Incident	0.987342358	1
CLE-M-2016,	RE	Runway Excursion	0.991727948	1
FCM-M-201	D	Category D RI	0.993170709	1
TPA-M-2015	SI	Surface Incident	0.995973051	1
SUS-M-2016	SI	Surface Incident	0.998512089	1
DTW-M-201	SI	Surface Incident	0.86172237	2
APF-M-2015	D	Category D RI	0.913455188	2
FPR-M-2016	D	Category D RI	0.917891892	2
ITH-M-2015/	D	Category D RI	0.922386457	2
DVT-M-2016	SI	Surface Incident	0.924336426	2
RHV-M-2016	D	Category D RI	0.934876069	2
BIL-M-2016/	RE	Runway Excursion	0.963671872	2
TEB-M-2015,	RE	Runway Excursion	0.966555952	2
LGB-M-2015	SI	Surface Incident	0.851610502	3
LAX-M-2016	D	Category D RI	0.851908115	3
MOD-M-201	D	Category D RI	0.853150891	3
MYF-M-2016	D	Category D RI	0.854466288	3
MDW-M-20	D	Category D RI	0.857430368	3
BUR-M-2016	SI	Surface Incident	0.858623675	3

Model Deployment

- **Deployed in two ways:**
  - Accessible from a web page
  - Through a web service

Developing Ansible playbook and launching from Ansible Tower:

Prepared By: Ramesh Adhikari

This tutorial was prepared based on experience during the deployment of SMC-API project from (ANG Labs, https://ansible.faa.gov) Ansible Tower. It includes all steps: from getting access of the resources to the setting up and launching for deployment.

I. Need access to the following:

1) Project git-repo (git.faa.gov/projects/"name-of-project": *Contact repo owner*
2) EIMEC git-repo (git.faa.gov/projects/EIMEC): *Contact repo owner*
3) Should be able to create the ssh url in the EIMEC repo: *Contact repo owner + Ansible tower team*
4) Target server (server to deploy the project): *Create a JIRA ticket*
5) Ansible tower (ansible.faa.gov)
   (*To get access, create the JIRA ticket from http://jira.faa.gov/. Remember, to get access to the inventory server in tower, need to submit separate ticket from the same link*)

II. Git-repo:

A) An Ansible playbook file 'main.yml' that defines the Ansible roles.

   Example:

```
- name: Installation and deployment for SMC-API
     hosts: all
     gather_facts: no
     become: false
     roles:
       - smc
```

   A main.yml defining Ansible roles paths "/roles/smc/ (tasks, files, defaults, vars, templates)".

B) 'roles' folder which may have many components.

   'roles' contains only a single subdirectory with a name that indicates the project to run. Sub-sub-directories can be added inside this sub-directory as needed. The allowed sub-sub-directories are: 'tasks', 'handlers', 'files', 'library', 'templates', 'vars', 'defaults' and 'meta'. In each of the sub-sub-directories, Ansible will look a main.yml file with relevant contents. ‘tasks’ contains the main Ansible playbook to be executed for the whole project.

(Note: separate files ‘hosts’, ‘ansible.cfg’ that are needed to run Ansible from command-lines are not needed while running the playbook from tower.)

C) All other folders and files that appeared in the repo are required by the project: from model development/training/scoring to deployment. Note that Ansible playbook will read the files from the same repository.

D) The Git-repo which is not publicly accessible, is required to make accessible from the Tower. It follows the specific process.

The ANG Labs of ansible.faa.gov has been connected to the git project repo named EIMEC. Before moving to Ansible tower dashboard to create the project, the current git-repo needs to be forked to the project repo git.faa.gov/projects/EIMEC and make the project accessible to clone with ssh-url (It won’t work with http-url).

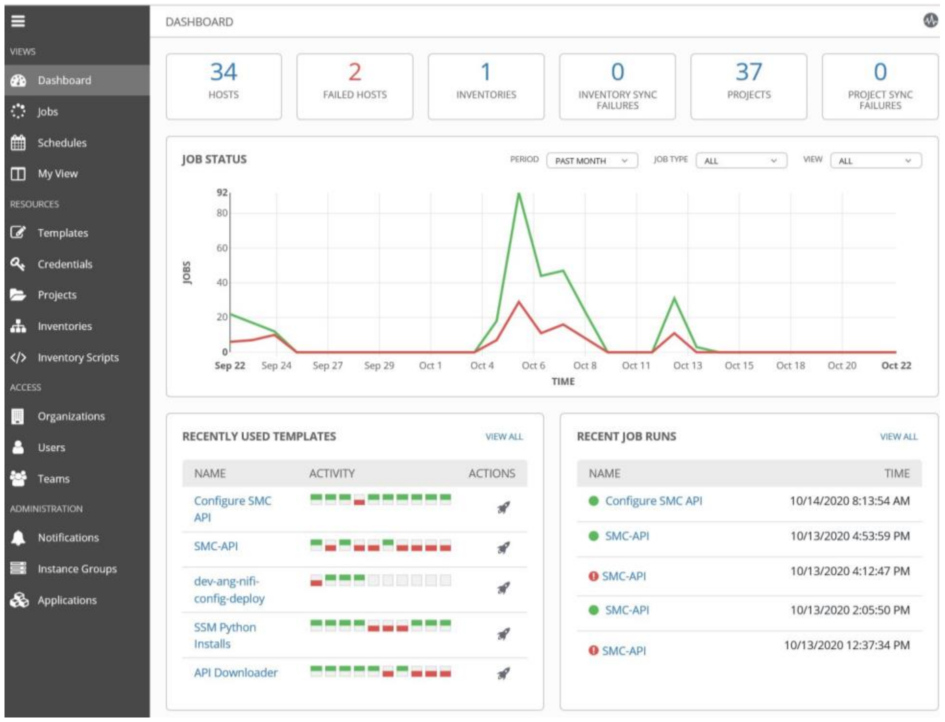
III. Ansible Tower:

A) Log into the Ansible tower browsing the link ‘ansible.faa.gov’

a) Type ansible.faa.gov on the address bar, it will take you to login page

b) At the bottom left of the page there is ‘Sign in with S’ button, click ‘S’ button.

c) If you already have permission, you will see the Ansible tower page with the menu bar in the left column (as in the screenshot shown below).



34	2	1	0	37	0
HOSTS	FAILED HOSTS	INVENTORIES	INVENTORY SYNC FAILURES	PROJECTS	PROJECT SYNC FAILURES

RECENTLY USED TEMPLATES

NAME	ACTIVITY	ACTIONS
Configure SMC API	🟩🟩🟩🟥🟩🟩🟩🟩🟩🟩	
SMC-API	🟩🟥🟩🟥🟥🟩🟥🟥🟥🟥	
dev-ang-nifi-config-deploy	🟥🟩🟩🟩	
SSM Python Installs	🟩🟩🟩🟩🟥🟥🟥🟩🟩🟩	
API Downloader	🟩🟩🟩🟩🟩🟥🟩🟥🟥🟥	

RECENT JOB RUNS

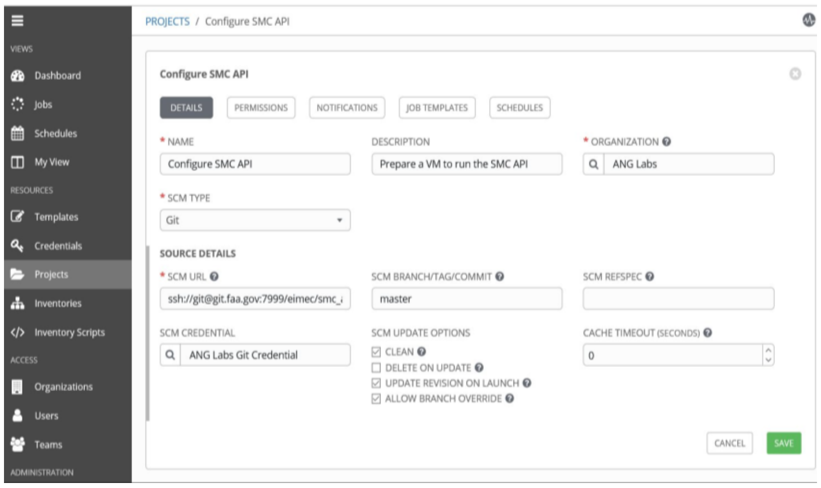
NAME	TIME
● Configure SMC API	10/14/2020 8:13:54 AM
● SMC-API	10/13/2020 4:53:59 PM
● SMC-API	10/13/2020 4:12:47 PM
● SMC-API	10/13/2020 2:05:50 PM
● SMC-API	10/13/2020 12:37:34 PM

B) It needs the full access to the tower. To check if you have full access or not:

- ➢ Click 'Organization' tab: it will show the registered organization under your account
- ➢ Click 'Inventory' tab: it will list the inventories registered under your account.
- ➢ Also, User tab: and check if you are allowed as admin

C) Create job project:

- ➢ Click the Projects tab:
- ➢ Click the green '+' button to add the project
- ➢ A window with boxes will be appeared. Fill the boxes as shown here in the screenshot below:



Configure SMC API

* NAME: Configure SMC API	DESCRIPTION: Prepare a VM to run the SMC API	* ORGANIZATION: ANG Labs
* SCM TYPE: Git		

SOURCE DETAILS

* SCM URL: ssh://git@git.faa.gov:7999/eimec/smc_	SCM BRANCH/TAG/COMMIT: master	SCM REFSPEC:
SCM CREDENTIAL: ANG Labs Git Credential	SCM UPDATE OPTIONS: ☑ CLEAN ☐ DELETE ON UPDATE ☑ UPDATE REVISION ON LAUNCH ☑ ALLOW BRANCH OVERRIDE	CACHE TIMEOUT (SECONDS): 0

a) NAME: Name of the Project, it can be anything relevant to the project
b) DESCRIPTION: (Optional) type anything relevant to the project or leave blank
c) ORGANIZATION: Name of the organization from the list of the organizations (example: ANG Labs)
d) SCM TYPE: choose Git from the drop down menu:

   After choosing Git, more boxes will be appeared.

e) SCM URL: Choose ssh-url that you created when project is cloned to the EIMEC repo as discussed in previous section of this tutorial.

   (Note: if the repo is publicly read accessible, http-url of the repo works but if it is not accessible to the public it must need ssh-url)

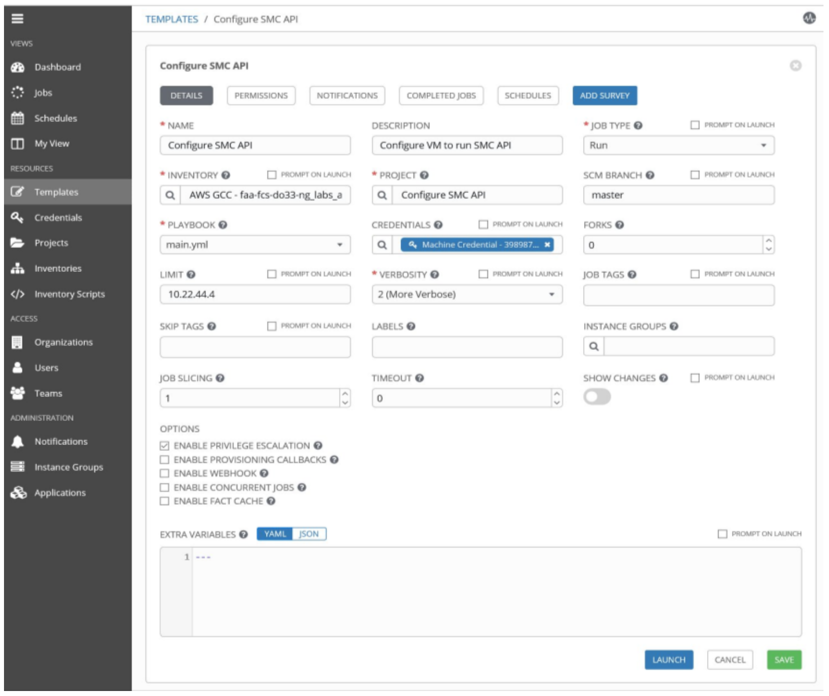
f) SCM BRANCH/TAG/COMMIT: The branch of the repo which is to be deployed.
g) SCM REFSPEC: (Optional) you can leave it blank if don't have any
h) SCM CREDENTIAL: Need to use the relevant credential, to run the project from git-repo there is source control type credential (ex. in the ANG Labs, 'ANG Labs Git Credential')
i) SCM UPDATE OPTIONS: (Optional) Can be chosen as required

j) CACHE TIMEOUT (SECONDS): 0

- ➢ Click SAVE
- ➢ Click the Project tab again, the project will be appeared in the list of projects with green bullet if project created correctly, and with red '!' symbol if the project creation failed (clicking the '!' symbol you can go to the error page). Click arrowed circle to check if it updates the credentials and the project is created successfully. Otherwise, go back to the project page and modify the entries in the boxes as needed.

D) Create job template:

- ➢ Click Template tab:
- ➢ Click the green '+' tab to add the template
- ➢ A window with several boxes will be appeared. Fill the boxes as in the screenshot below:



TEMPLATES / Configure SMC API

Configure SMC API

* NAME: Configure SMC API	DESCRIPTION: Configure VM to run SMC API	* JOB TYPE: Run [ ] PROMPT ON LAUNCH
* INVENTORY: AWS GCC - faa-fcs-do33-ng_labs_a [ ] PROMPT ON LAUNCH	* PROJECT: Configure SMC API	SCM BRANCH: master [ ] PROMPT ON LAUNCH
* PLAYBOOK: main.yml	CREDENTIALS: Machine Credential - 398987... [ ] PROMPT ON LAUNCH	FORKS: 0
LIMIT: 10.22.44.4 [ ] PROMPT ON LAUNCH	* VERBOSITY: 2 (More Verbose) [ ] PROMPT ON LAUNCH	JOB TAGS: [ ] PROMPT ON LAUNCH
SKIP TAGS: [ ] PROMPT ON LAUNCH	LABELS:	INSTANCE GROUPS:
JOB SLICING: 1	TIMEOUT: 0	SHOW CHANGES: [ ] PROMPT ON LAUNCH

OPTIONS
- [✓] ENABLE PRIVILEGE ESCALATION
- [ ] ENABLE PROVISIONING CALLBACKS
- [ ] ENABLE WEBHOOK
- [ ] ENABLE CONCURRENT JOBS
- [ ] ENABLE FACT CACHE

EXTRA VARIABLES [✓ YAML] JSON [ ] PROMPT ON LAUNCH

```
---
```

a) NAME: Name of the job template can be same as project name.

b) INVENTORY: Name of the inventory listed in your inventories list

c) DESCRIPTION: It can be anything that describe the project
d) JOB TYPE: Choose 'Run'
e) PROJECT: Name of the project that was recently created and to be run
f) SCM BRANCH: specify the branch of the git-repo, same branch as used when creating the project
g) PLAYBOOK: Name of the Ansible playbook .yml file located in the tasks folder to be executed (example: main.yml)
h) CREDENTIALS: Machine credentials which is already set for the Organization, leave the default.
i) LIMIT: IP address of the target server where the project to be deployed.
j) VERBOSITY: it is optional, for complete debug choose 3.
k) JOB TAGS: Optional, leave blank if not available
l) SKIP TAGS: Optional, leave blank if not available
m) LABELS: Optional, leave blank if not available
n) INSTANCE GROUPS: Optional, leave blank if not available
o) JOB SLICING: 1
p) TIMEOUT: 0
q) SHOW CHANGES: Turn on if you want to see the changes from the previous run.
r) OPTIONS: Check the item 'ENABLE PRIVILEGE ESCALATION'

- Click SAVE

E) Run the Ansible Job from the tower.

- Click LAUNCH (in the Template window)

If everything is correct it will run the Ansible script stored inside the 'tasks' sub-sub-directory of the project repo in the git. It will follow all steps of installations and deployment into the target server sequentially as provided in the script.

Is the project deployed?

When the job completes successfully from the tower, open web browser and type http://(IP-address-of-the-target-server).

If the project is successfully deployed, it will show a page with template to upload the input file. Browse the input file and hit 'Submit' and wait for output.

It is possible that the project has not been deployed even if the Ansible tower job ended as 'Successful'. In such case, check the error.

To see the error-log:

- Go to terminal, ssh the server with access-user and password to it.
- Type 'sudo vi /var/log/httpd/error_log' and hit enter and scroll down to the bottom line to see the most recent error. Go back to script and fix the problem if any.

IV. Ansible playbook:

This section guides how to create the Ansible playbook for the deployment of a data science project. (***Following steps are required to run the project located in git-repo from the Ansible tower templates. To run from command-line some modifications are needed.***)

Ansible playbook is a simple script written in yml language. The first line contains three dash (---). Any line started with '#' will be ignored in the script. Ansible modules and instruction to implement the modules are well explained in the Ansible documentation https://docs.ansible.com/ansible/latest/index.html. In this document, you can find a guidance to deploy a specific project and some important facts to remember while preparing a playbook for that purpose. Also, this tutorial assumes that the codes in the project to be deployed in a bare server is written in python scripting language.

1) Write up full instruction to install apache server, correct version of python and required packages on current version of RHEL.
   Example: README page on
   https://git.faa.gov/projects/EIMEC/repos/smc_api/browse
2) Follow each step in the Ansible playbook.

Playbook Steps:

'become' module allows the user to operate the module if selected 'true'

To install: use module 'yum'

After colon of each module, there must be one 'space'

Variable must be placed in the format "{{ variable }}" (remember quotes, double braces, and single spaces before and after the variable.

Steps to be followed:

1) Ensure yum is updated
2) Install apache packages
3) Install apache-devel packages
4) Ensure the apache is running with module 'service' state 'started'
5) Ensure firewall port 80 is open with module 'firewalld'
6) Restart firewalld with module 'service' state 'restarted'
7) Install all pre-requisites for installing python3.7.7 (specifically required to the project)
   a. gcc.x86_64
   b. openssl-devel
   c. zlib-devel.x86_64
   d. bzip2-devel
   e. libgomp
   f. git
   g. patch
   h. libffi-devel
   i. sqlite.x86_64, requires 'update_cache'
   j. sqlite-devel.x86_64, requires 'update_cache'

k. ncurses, requires ‘update_cache’
l. xz-devel.x86_64, requires ‘update_cache’
m. libuuid-devel, requires ‘update_cache’

Note: Install all libraries one at a time to avoid interferences.

8) Follow steps to install python3.7.7, all steps requires module ‘tags: packages, python’ except step e. and f.
   a. It downloads the correct version of python using module ‘unarchive’. Important: src: `https://www.python.org/ftp/python/3.7.7/Python-3.7.7.tar.xz` and dest: /tmp/, and remote_src: yes
   b. Configure with module ‘shell: ./configure –enable-shared chdir=/tmp/Python-3.7.7’
   c. Make with module ‘shell: make chdir=/tmp/Python-3.7.7’
   d. Install with module ‘shell: make install chdir=/tmp/Python-3.7.7’
   e. Set conf for python3.7.7 with module ‘command: `bash -c 'echo "/usr/local/lib" > /etc/ld.so.conf.d/python-3.7.conf`’
   f. Reload libraries with module ‘command: ldconfig’
   g. Remove tmp files used for python installation with module ‘file: path= file path to the files to be removed and state=absent’
9) Upgrade pip using module ‘command: ‘
10) Create virtual environment and install packages from requirements.txt:
   a. a directory set accessibility using “mode: 0755”
   b. create virtual environment using module “command: ”
   c. Create a directory (with mode: 0755) to store the requirements.txt file.

      Note: installation of requirements in virtualenv demands the requirements.txt must be stored in the target server, not in the git-repo.
   d. Copy the requirements.txt from the project repo in git to the recently created directory.
   e. Pip install python packages using module ‘pip: ‘ with sub-modules ‘requirements, virtualenv, and virtualenv_python’

      Note: This requires virtualenv and setuptools as pre-requisites although already installed with installation of python in most of the python versions.

11) Prepare WSGI
    a. Create WSGI directory /var/www/wsgi with mode: 0755
    b. Copy required files/folders to the recently created wsgi directory
    c. Create a directory within wsgi (with mode: 0755) to store project specific wsgi_.py file.
    d. Copy that wsgi_.py file to that newly created directory within wsgi
    e. Copy .ini file stored in /roles/…/files/ which contains path to .html templates file and folders in step 11) b.
    f. Set root permission (group and owner) on wsgi directory
    g. Set restorecon and setsebool with module "command: "
12) Patch the files if needed, using module "patch: "
    In the project under example, ctypes.patch is required to be invoked.
13) Copy project specific httpd.conf stored in /roles/../files/ to the directory /etc/httpd/conf.d/
14) At the end of the playbook, make sure httpd is running (same as in step 4 but with 'state: restarted')

V. Configure SSL connection:

The above procedures complete the deployment of the model in a bare server through port 80. Template for uploading input file and submit button should be appeared upon browsing the http://<server-ip-address>. To make the connection secured, one needs to follow steps:

1) Get CA and chain certificates: (*If provided with private key and corresponding certificate files in the .key and .crt format skip this and move to the next step*)
    - ♦ ssh log into the server where the model to be deployed.
    - ♦ Make sure the server contains the previously installed openssl.
    - ♦ To get the ssl CA certificates from the trusted CA authority, create a 'san.conf' file (with the IP address of the target server):

```
[ req ]
default_bits = 2048
distinguished_name = req_distinguished_name
req_extensions     = req_ext
```

```
[ req_distinguished_name ]
countryName = Country Name (2 letter code)
stateOrProvinceName = State or Province Name (full
name)
localityName = Locality Name (eg, city)
organizationName = Organization Name (eg, company)
commonName = Common Name (e.g. server FQDN or YOUR
name)
[ req_ext ]
subjectAltName = @alt_names
[ alt_names ]
IP.1     = 10.22.44.4
```

- Enter the following command in the server for which the CA certificate is needed:

```
openssl req -out sslcert.csr -newkey rsa:2048 -
nodes -keyout private.key -config san.conf
```

- It will prompt to the fields to be filled and creates the private key file private.key and CA certificate request file sslcert.csr:

```
Generating a 2048 bit RSA private key
...................................................
................................................+++
...............+++
writing new private key to 'private.key'
-----
You are about to be asked to enter information that
will be incorporated
into your certificate request.
What you are about to enter is what is called a
Distinguished Name or a DN.
There are quite a few fields but you can leave some
blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) []: US
```

```
State or Province Name (full name) []: NJ
Locality Name (eg, city) []: Atlantic City
Organization Name (eg, company) []: FAA
Common Name (e.g. server FQDN or YOUR name) []: ip-
10-22-44-4.fcs.faa.gov
```

(*Hint: If you don't know FQDN, ssh the server and type 'hostname' and hit ENTER, it will result the FQDN.*)

- After getting .csr file, navigate to: https://iamcdcpvap110.amc.faa.gov/certsrv/ (must use Internet Explorer, not Chrome)
- Click "Request a certificate"
- Click "Advanced certificate request"
- Click "Submit a certificate request by ……………………………………… "
- Paste contents of .csr file created in above step (Remember after copy and paste there should be one blank line at the end of the field)
- Type your faa-email address in the field 'Govt POC Email'
- Click 'submit' button
- Check 'Base 64 encoded'
- Click 'Download certificate': it will download the certificate file as 'certnew.cer'
- Click 'Download certificate chain': it will download the certificate chain file as 'certnew.p7b'
- Collect the .key, .crt and .p7b files and store it to the 'roles/smc/files' folder
- Rename private.key to <FQDN>.key
- Convert certnew.cer to <FQDN>.crt and certnew.p7b to <FQDN>.crt as:
  - openssl x509 -inform PEM -in certnew.cer -out <FQDN>.crt
  - openssl pkcs7 -print_certs -in certnew.p7b -out <FQDN>.chain.crt

  (Note I: Named .key and .crt files according to the FQDN 'ip-10-22-44-4' of the server node)

  (Note II: key and crt files can be encrypted using ansible-vault if it is accessible from the ansible tower)

2) Prepare the ssl configuration file:

For preparing the ssl configuration file, create the 'templates' folder in 'roles/smc' and then create the configuration file as:

```
LoadModule ssl_module /usr/lib64/httpd/modules/mod_ssl.so

# Listen on port
Listen {{ httpd_listen_ssl }} https

## SSL Global Context
SSLPassPhraseDialog exec:/usr/libexec/httpd-ssl-pass-dialog
SSLSessionCache shmcb:/run/httpd/sslcache(512000)
SSLSessionCacheTimeout  300
SSLRandomSeed startup file:/dev/urandom  256
SSLRandomSeed connect builtin
SSLCryptoDevice builtin

## SSL Virtual Host Context
<VirtualHost 10.22.44.4:{{ httpd_listen_ssl }}>

SSLEngine on

ErrorLog {{ httpd_error_log_ssl }}
TransferLog {{ httpd_access_log }}
LogLevel {{ httpd_log_level_ssl }}

## Certificate files
SSLCertificateFile {{ httpd_cert_dir }}/{{ ssl_cert }}
SSLCertificateKeyFile {{ httpd_key_dir }}/{{ ssl_key }}
SSLCACertificateFile {{ httpd_cert_dir }}/{{ ssl_ca_cert }}

<Files ~ "\.(cgi|shtml|phtml|php3?)$">
    SSLOptions +StdEnvVars
</Files>
<Directory "/var/www/cgi-bin">
    SSLOptions +StdEnvVars
</Directory>

BrowserMatch "MSIE [2-5]" \
         nokeepalive ssl-unclean-shutdown \
         downgrade-1.0 force-response-1.0

CustomLog logs/ssl_request_log \
          "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"

</VirtualHost>

## SSL Configuration
## see https://mozilla.github.io/server-side-tls/ssl-config-generator/
SSLProtocol {{ httpd_ssl_protocol }}
SSLCipherSuite {{ httpd_ssl_cipher_suite }}
SSLHonorCipherOrder {{ httpd_ssl_honor_cipher_order }}
SSLCompression {{ httpd_ssl_compression }}
SSLSessionTickets {{ httpd_ssl_session_tickets }}

## OCSP Stapling, only in httpd 2.3.3 and later
SSLUseStapling {{ httpd_ssl_use_stapling }}
SSLStaplingResponderTimeout {{ httpd_ssl_stapling_responder_timeout }}
SSLStaplingReturnResponderErrors {{ httpd_ssl_stapling_return_responder_errors }}

SSLStaplingCache {{ httpd_ssl_stapling_cache }}
```

{*Variables inside* {{ }} *are given in the 'defaults/main.yml' or in the 'vars/main.yml' files.*}

3) Install package mod_ssl together along with httpd and install the private key, ca-certificate and chain-certificate in the server.
    a. Prepare a separate ‘new_cert.yml’ file for installing and configuring ssl certificate in the server.
    b. Add a block in the ansible playbook ‘main.yml’ to include the ‘new_cert.yml’ using ‘include’ module.
    c. new_cert.yml contains the blocks to:
        - Modify the ssl.conf file by replacing the default name of the key and self-signed certificate files ‘localhost.key’ and ‘localhost.crt’ to ‘<FQDN>.key’ and ‘<FQDN>.crt’, respectively.
        - Copy and save those key and crt files from ‘roles/smc/files/’ to ‘/etc/pki/tls/private’ and ‘etc/pki/tls/certs/’ of the server respectively.
        - Copy and save the ssl.conf.j2 file from ‘roles/smc/templates/’ to ‘/etc/httpd/conf.d/’ of the server.

4) Install the CA certificate in the web browsers (ex. Chrome, Firefox, Microsoft Edge, etc.)
    - In the windows, open ‘command prompt’ using search bar.
    - Type ‘certmgr.msc’ and follow
        - Go to the ‘trusted ca certificate …….’
        - Right click the CA certificate and follow the steps until the successful installation of the certificate.
    - Or, open the browser, go to the setting from the right corner of the address bar, go to the advanced setting, security management and follow the instruction to install the certificates in the corresponding browser.

5) Now, type the https://<server-ip-address> and hit Enter, it will open the webpage with the template same as in http://<server-ip-address>.

VI. To deploy in EIM DEV:

All the above steps are for deploying the model in Ang Labs. It needs more steps to deploy in EIM DEV and one needs extra access to create projects and templates in the tower (or someone EIM team can help to create the project and template in tower dashboard). In addition, one needs to follow the steps:

a. get server IP/ available security port
b. make sure that firewall is on
c. Create the new project according to the new server in EIM
d. Rename the server specific values of the variables in roles/smc/defaults/main.yml
e. Change the values of "servername" and "httpd_listen_ssl" with appropriate IP-address and listening port number.
f. Store the server specific ssl certificates and key files with appropriate names in the folder roles/smc/files/
g. Update the changes to the git-repo
h. Go to the Ansible tower dashboard
i. Create new project with EIM credentials
j. Create new templates according to the new project
k. Hit the 'SAVE' and 'LAUNCH' button

Important: new port may not be added to the firewall list, to see if it is in the firewall list or not, follow the steps:

- ssh the server
- make sure that firewalld is running by using command 'systemctl status firewalld'
- make sure that httpd is running by using command 'systemctl status httpd.service'
- type command 'sudo firewall-cmd --list-all' and hit ENTER
- It will show the list of opening ports under the topic "ports:"
- If the desired port is not listed, add it using command: "sudo firewall-cmd --zone=public --permanent --add-port=11443/tcp" (use appropriate port number, here we used 11443)

- Check if added using the command ‘sudo firewall-cmd --list-all’
- It needs to restart firewalld and httpd to see the added port in the list.

VII. Tests:

For this particular app,

1) Web-browser:

   Different input formats can be expected successfully run. In this project, three .xls, .xlsx and .csv formats of inputs are allowed to upload in the browser as described above.

2) Restful End:

   Input in the .json format can be run through restful end using the command in terminal: `curl -s -d @JSON-input1.json -H "Content-Type: application/json" https://<IP-Address>:<port>/score_json`

   where, `JSON-input1.json` is the input in .json format

   <port> is the port used in section VII (if it is different from standard port 443)

   “<IP-Address>:<port>” can be replaced by the web-link if available.

VIII. Ansible roles files in the Git-Repo:

The Ansible playbook developed in the above section is to be stored in the directory roles/…name-of-roles.../tasks/. As already mentioned in section **Git-repo**, there will be other sub-directories together with ‘tasks’ which are ‘files’, ‘defaults’, ‘vars’, ‘templates’ as needed. We have following sub-directory structure inside the Ansible roles.

- roles:
  - smc:
    - defaults:
      - main.yml
    - files:
      - <domain_name>.key
      - <domain_name>.crt
      - <domain_name>.chain.crt

            - smc_httpd.conf
            - smor.ini
        - tasks:
            - cert_main.yml
            - main.yml
        - templates:
            - ssl.conf.j2
        - vars:
            - main.yml

(**IMPORTANT**: If ‘defaults, vars, templates, tasks, files’ are present inside the roles/smc/ then other files/folders in the repo should not be named as these names because Ansible playbook sees those names from the roles directory first and works only on those.)