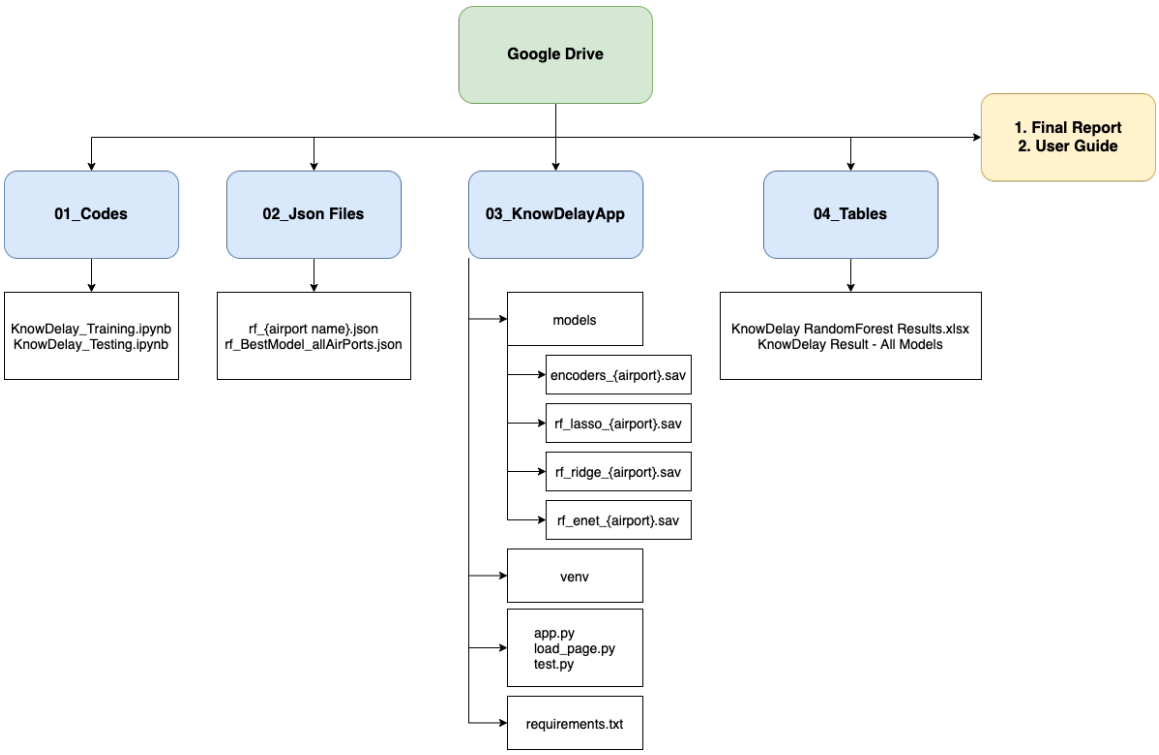
Know Delay - Team D: User Guidelines

This document is to provide instructions on executing codes for retraining/testing purposes, along with guideline to executing API based app for making prediction. It also consists of folder structure and elaboration of each folder.

1. Folder Structure



2. Contents

01_Codes

Please note that in order for these scripts to properly function, the input file needs to be in exactly the same format as that of trained files. In other words, they need to be xlsx files with the 'Data' tab, also column B to X need to be present with the same header names.

KnowDelay_Training.ipynb

This script should be utilized for retraining purposes. The file is constructed in the Jupyter Notebook Python platform, which consists of multiple functions that process raw data and train it, along with saving model and json files. To execute, all cells need to be run.

Input directory, where raw data is stored, need to be changed accordingly in function 'combine_files.'

```
import os
import datetime as dt
def combine_files():
    filePath= '/Users/jinhaenglee/Desktop/Georgia_Tech_OMSA/Practicum/Data/'
    i=0
    dct={}
    for file in os.listdir(filePath):
        f=file
        if f.endswith('.xlsx'):
            tmpDF=pd.read_excel(open(filePath+'/'+f, 'rb'),sheet_name='Data',header=1)
            tmpDF['airportNM']=f[0:3]
            dct[f]=tmpDF

    df_lst=[]
    for k,v in dct.items():
        df_lst.append(v)
    dataDF= pd.concat(df_lst)
    dataDF.rename(columns={dataDF.columns[15]: 'WXCodes'},inplace=True)
    dataDF.rename(columns={dataDF.columns[22]: 'SchedDepart'},inplace=True)
    dataDF.rename(columns={dataDF.columns[23]: 'SchedArriv'},inplace=True)

    return dataDF
```

By default, the script is looping through all airports. If a user wants to selectively loop through certain airports, the function 'train' needs to be manually changed. Flag 0 is for all airports, 1 for selected airports. These airports need to be changed as a list.

```
def train(flag):
    param_grid = {'bootstrap': [True],
                  'max_depth': [50,70,100],
                   'max_features': [5],
                   'min_samples_leaf': [3],
                   'min_samples_split': [8],
                   'n_estimators': [50, 100, 200]}


    dataDF = combine_files()
    if flag == 0:
      airports=list(dataDF['airportNM'].unique())
    elif flag == 1:
      airports = ['BWI', 'CLE']


    for airport in airports:
        print('Starting data prep for',airport)
        airportdf=dataDF[(dataDF['airportNM']==airport)]
        train_X, test_X, train_y, test_y = dataprep(airportdf,airport)
        print('Finished data prep for',airport)
        models_dct = RandomForest(train_X, train_y,test_X,test_y,param_grid,airport)
        finalModels = SelectBestToJson(models_dct, airport)
        print('Model training completed')


        filename = 'rf_{}.json'.format(airport)
        with open(filename, "w") as outfile:
            json.dump(finalModels, outfile)
```

In the above case, 'BWI' and 'CLE' are selected. In the 'main' function, flag needs to be called to be either 0 or 1.

```
def main():
    trainFlg='Y'
    if trainFlg=='Y':
        train(1)

if __name__ == "__main__":
    #weatherDataDF=combine_files()
    main()
```

KnowDelay_Testing.ipynb

This script should be utilized when a user wants to test the model with a dataset, whether it is for all flights or for delayed flights only. Similarly, to the training script, running all cells in this Jupyter Notebook will execute testing.

Function 'test' needs to be tailored around for a user. First, there are four directories: 1. Model directory where all models are stored, 2. Encoder directory where all encoders are stored, 3. Input directory, where raw data (xlsx files) are stored, and 4. Output directory, where the final csv file will be stored (highlighted in red below).

Additionally, the path of main json file (rf_BestModel_allAirPorts.json) needs to be specified (highlighted in blue below)

```
def test(flag):
    jsonFile = '/Users/jinhaenglee/Downloads/rf_BestModel_allAirPorts.json'
    with open(jsonFile) as f:
      data = json.load(f)

    models = list(data.keys())
    modelsDir = '/Users/jinhaenglee/Downloads/Models/'
    encodersDir = '/Users/jinhaenglee/Downloads/Models/Encoders/'
    inputDir = '/Users/jinhaenglee/Desktop/Georgia_Tech_OMSA/Practicum/Data/'
    outputDir = '/Users/jinhaenglee/Downloads/'
    files = glob.glob(inputDir + '*.xlsx')
    airports = []
    for f in files:
        airports.append(f.split('/')[-1].split(' ')[0])
```

Moreover, when running the main function, a user needs to specify a flag input with test function, when it is called. Flag == 0 to test the model on test dataset, flag == 1 for test dataset with delayed flights only. As we bucketed delay minutes into 10-minute interval, filtering here is 3 for 30-minute threshold. If a user wishes to use different threshold, this number needs to be tuned (See screenshot below).

```
[ ]  def main():
        testFlag = 'Y'
        if testFlag == 'Y':
            test(1)

    if __name__ == "__main__":
        main()
```

02_Json Files

Json Files folder contains json files generated for each airport and a combined version for all airports. Each json file is structured as follows:

{model file name:
{airport: airport name,
features: features selected for this airport
accuracy: accuracy calculated for this model
LL Delayed: Likelihood of being delayed}
}

Combined version json file, 'rf_BestModel_allAirPorts.json' is also located inside KnowDelay App folder that will be called as input.

03_KnowDelayApp

KnowDelayApp folder is the package folder containing required codes, texts, and input files for API application.

I. Model folder:

Model folder contains .sav extension files that are required input for API call.

- **Encoders**: binary encoder that needs to be loaded to recall consistent features.
- **Lasso**: Lasso feature selection method result for each airport. Selected features are saved that can be recalled back from the json file.
- **Ridge**: Ridge feature selection method result for each airport. Selected features are saved that can be recalled back from the json file.
- **Enet**: Elastic Net feature selection method result for each airport. Selected features are saved that can be recalled back from the json files.

II. Venv folder:

This is a lightweight virtual environment set up for API call

III. Py extension files:

Necessary codes for API call/response. Test.py is the main executing code for the user.

IV. Requirement.txt

Lists of packages or libraries required in order to run test.py script.

V. json files

Required input json files are saved here to be called once main script is executed.

04_Tables

'Tables' folder contains two spreadsheets.

- **KnowDelay RandomForest Results:** Contains two sub-tables; one with calculated accuracy, recall, precision, f1 score, and balanced accuracy for each airport. Note that this table is a result of Random Forest for non-delayed and delayed flights. Second table consists of the same metric calculation but with delayed flights only.
- **KnowDelay Results - All Models:** Unlike the above table, this is a result of 10-fold cross-validation for Random Forest, XGBoosting and K-nearest Neighbor modeling. It consists of which feature selection performed the best, as well as metrics calculated for each model. Moreover, likelihood for a flight to be delayed or not are included as well. Note that this is the source where plots are generated in the report.

3. Integration and Application

We present API integrated app. The package is included in the google drive folder called '03_KnowDelayApp.'

The single most important script in the execution phase is 'test.py' and it takes in two major inputs to operate properly.

First, the BASE URL (highlighted in red), needs to be specified. Currently, it is defined as a local hosting address but this needs to be changed depending on a user's hosting URL, where 'app.py' will be hosted to load and run prediction.

Second, testing data needs to be filled (highlighted in blue). It takes a form of dictionary, consists of features of the raw data. This is designed to extract columns from B to X in the provided raw data excel sheets, with addition of airport name.

Once above two inputs are tailored around user, 'app.py' script will take the input, searches and loads the best performing model with provided json file, make prediction and return the results.

```
import requests
import json

BASE="http://127.0.0.1:5000/"

data={"Airport":'SEA',
      "Temp":25,
      "Dew":39,
      "T/D Spread":5.2,
      "Wind Direction":20,
      "Wind Speed":12,
      "Wind Gust":18,
      "Altimeter":29.62,
      "Vis":0.5,
      "Sky 1":"FEW",
      "Sky 2":"BKN",
      "Sky 3":"BKN",
      "Layer 1":800,
      "Layer 2":3000,
      "Layer 3":8000,
      "WXCodes":'SN BLSN',
      "Ice 1 Hour":None,
      "Ice 3 Hour":None,
      "Ice 6 Hour":None,
      "Peak Gust Speed":None,
      "Peak Gust Direction": None,
      "Feels Like":66.9,
      "SchedDepart": 36,
      "SchedArriv":22
      }
```