Airline Delay Prediction using PySpark, PandasUDF and Scikit-Learn

In [1]:

```
from sklearn.datasets import make_classification
import pandas as pd
import numpy as np
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from pyspark.sql.types import StructType, StructField, IntegerType, FloatType
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, precision_score, recall_score, roc_auc_score
from pyspark.sql.functions import pandas_udf, struct, PandasUDFType
from sklearn.svm import SVC, LinearSVC
from sklearn.linear_model import LogisticRegression
```

In [2]:

```
df=pd.read_csv('C:/Users/Devanshi/Desktop/All recent required folders/Airline Delay Prediction/data/All origins.csv')
```

In [3]:

```
df['DELAYED'] = np.where(df['DEP_DELAY']> 10, 1, 0)
df = df.drop(['YEAR','ORIGIN_CITY_NAME','ORIGIN','DEST','ORIGIN_STATE_NM','CANCELLED','DEST_CITY_NAME',
    'DEST_STATE_NM', 'DEP_TIME','DEP_DELAY', 'ARR_TIME', 'ARR_DELAY'],axis=1)
```

In [4]:

```
df.columns
```

Out[4]:

```
Index(['MONTH', 'DAY_OF_MONTH', 'DAY_OF_WEEK', 'DISTANCE', 'CARRIER_DELAY',
    'WEATHER_DELAY', 'NAS_DELAY', 'SECURITY_DELAY', 'LATE_AIRCRAFT_DELAY',
    'id', 'DELAYED'],
   dtype='object')
```

In [5]:

```
schema = StructType([
StructField('id', IntegerType()),
StructField('recall', FloatType()),
StructField('precision', FloatType()),
StructField('accuracy', FloatType()),
StructField('auc', FloatType())
])
X_columns = df.drop(columns = ['id', 'DELAYED']).columns
y_columns = 'DELAYED'
```

In [6]:

```
df_spark = spark.createDataFrame(df)
```

Random Forest Algorithm

In [7]:

```
@pandas_udf(schema, PandasUDFType.GROUPED_MAP)
def model_results_per_id_rf(df):
    id = int(df.id.unique()[0])
    X = df[X_columns]
    y = df[y_columns]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)
    steps = [('scaler', StandardScaler()),
    ('rf', RandomForestClassifier(random_state=0, n_jobs=-1))
    ]
    pipeline = Pipeline(steps)
    param_distributions = {'rf__n_estimators': [50,100],'rf__min_samples_leaf': [1, 2, 4],'rf__max_depth': [5,10,20]}
    rf_cv = RandomizedSearchCV(pipeline, param_distributions, cv = 5, n_jobs = -1, scoring = 'f1')
    rf_cv.fit(X_train, y_train)
    y_pred = rf_cv.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred).tolist()
    precision = precision_score(y_test, y_pred).tolist()
    recall = recall_score(y_test, y_pred).tolist()
    y_pred_prob = rf_cv.predict_proba(X_test)[:,1]
    auc = roc_auc_score(y_test, y_pred_prob).tolist()
    model_results = pd.DataFrame([[id, recall, precision, accuracy, auc]], columns = ['id', 'recall', 'precision', 'accuracy', 'auc'])
    return model_results
```

In [8]:

```
%%time
model_results_by_id1 = df_spark.groupBy('id').apply(model_results_per_id_rf).toPandas()
```

```
C:\opt\spark\spark-3.0.1-bin-hadoop2.7\python\pyspark\sql\pandas\group_ops.py:73: UserWarning: It is preferred to use 'applyInPandas' over this A
PI. This API will be deprecated in the future releases. See SPARK-28264 for more details.
  warnings.warn(
```

```
Wall time: 7min 50s
```

In [9]:

```
%%time
model_results_by_id1[['recall', 'precision', 'accuracy', 'auc']] = model_results_by_id1[['recall', 'precision', 'accuracy', 'auc']].round(3)
model_results_by_id1.sort_values(by = 'id').reset_index(drop = True).head()
```

```
Wall time: 11.2 ms
```

Out[9]:

	id	recall	precision	accuracy	auc
0	1	0.650	0.979	0.927	0.869
1	2	0.728	0.973	0.945	0.888
2	3	0.692	0.987	0.931	0.884
3	4	0.714	0.973	0.936	0.894

Support Vector Machine

In [10]:

```
from sklearn.calibration import CalibratedClassifierCV
```

In [11]:

```
@pandas_udf(schema, PandasUDFType.GROUPED_MAP)
def model_results_per_id_svm1(df):
    id = int(df.id.unique()[0])
    X = df[X_columns]
    y = df[y_columns]
    X_train, X_test, y_train, y_test = train_test_split(X, y,random_state = 0)
    svm = LinearSVC(C=5, random_state = 67)
    clf = CalibratedClassifierCV(svm)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred).tolist()
    precision = precision_score(y_test, y_pred).tolist()
    recall = recall_score(y_test, y_pred).tolist()
    y_pred_prob = clf.predict_proba(X_test)[:,1]
    auc = roc_auc_score(y_test, y_pred_prob).tolist()
    model_results = pd.DataFrame([[id, recall, precision, accuracy, auc]], columns = ['id', 'recall', 'precision', 'accuracy', 'auc'])
    return model_results
```

In [12]:

```
%%time
model_results_by_id2a = df_spark.groupBy('id').apply(model_results_per_id_svm1).toPandas()
```

```
C:\opt\spark\spark-3.0.1-bin-hadoop2.7\python\pyspark\sql\pandas\group_ops.py:73: UserWarning: It is preferred to use 'applyInPandas' over this A
```

```
C:\opt\spark\spark-3.0.1-bin-hadoop2.7\python\pyspark\sql\pandas\group_ops.py:73: UserWarning: It is preferred to use 'applyInPandas' over this A
PI. This API will be deprecated in the future releases. See SPARK-28264 for more details.
  warnings.warn(
```

Wall time: 4min 46s

In [13]:

```
%%time
model_results_by_id2a[['recall', 'precision', 'accuracy', 'auc']] = model_results_by_id2a[['recall', 'precision', 'accuracy', 'auc']].round(3)
model_results_by_id2a.sort_values(by = 'id').reset_index(drop = True).head()
```

Wall time: 0 ns

Out[13]:

	id	recall	precision	accuracy	auc
0	1	0.174	0.999	0.835	0.819
1	2	0.268	0.999	0.864	0.863
2	3	0.300	1.000	0.847	0.828
3	4	0.231	1.000	0.837	0.858

In [14]:

```
@pandas_udf(schema, PandasUDFType.GROUPED_MAP)
def model_results_per_id_svm2(df):
    id = int(df.id.unique()[0])
    X = df[X_columns]
    y = df[y_columns]
    X_train, X_test, y_train, y_test = train_test_split(X, y,random_state = 0)
    svm = LinearSVC(C=5, random_state = 67)
    clf = CalibratedClassifierCV(svm, method='isotonic')
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred).tolist()
    precision = precision_score(y_test, y_pred).tolist()
    recall = recall_score(y_test, y_pred).tolist()
    y_pred_prob = clf.predict_proba(X_test)[:,1]
    auc = roc_auc_score(y_test, y_pred_prob).tolist()
    model_results = pd.DataFrame([[id, recall, precision, accuracy, auc]], columns = ['id', 'recall', 'precision', 'accuracy', 'auc'])
    return model_results
```

In [15]:

```
%%time
model_results_by_id2b = df_spark.groupBy('id').apply(model_results_per_id_svm2).toPandas()
```

```
C:\opt\spark\spark-3.0.1-bin-hadoop2.7\python\pyspark\sql\pandas\group_ops.py:73: UserWarning: It is preferred to use 'applyInPandas' over this A
PI. This API will be deprecated in the future releases. See SPARK-28264 for more details.
  warnings.warn(
```

Wall time: 4min 45s

In [16]:

```
%%time
model_results_by_id2b[['recall', 'precision', 'accuracy', 'auc']] = model_results_by_id2b[['recall', 'precision', 'accuracy', 'auc']].round(3)
model_results_by_id2b.sort_values(by = 'id').reset_index(drop = True).head()
```

Wall time: 3.96 ms

Out[16]:

	id	recall	precision	accuracy	auc
0	1	0.642	0.972	0.925	0.836
1	2	0.720	0.958	0.942	0.869
2	3	0.675	0.979	0.926	0.849
3	4	0.709	0.952	0.931	0.864

Logistic Regression

In [17]:

```
@pandas_udf(schema, PandasUDFType.GROUPED_MAP)
def model_results_per_id_lr(df):
    id = int(df.id.unique()[0])
    X = df[X_columns]
    y = df[y_columns]
    X_train, X_test, y_train, y_test = train_test_split(X, y,random_state = 0)
    clf = LogisticRegression(max_iter=1000, C=100).fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred).tolist()
    precision = precision_score(y_test, y_pred).tolist()
    recall = recall_score(y_test, y_pred).tolist()
    y_pred_prob = clf.predict_proba(X_test)[:,1]
    auc = roc_auc_score(y_test, y_pred_prob).tolist()
    model_results = pd.DataFrame([[id, recall, precision, accuracy, auc]], columns = ['id', 'recall', 'precision', 'accuracy', 'auc'])
    return model_results
```

In [18]:

```
%%time
model_results_by_id3 = df_spark.groupBy('id').apply(model_results_per_id_lr).toPandas()
```

```
C:\opt\spark\spark-3.0.1-bin-hadoop2.7\python\pyspark\sql\pandas\group_ops.py:73: UserWarning: It is preferred to use 'applyInPandas' over this A
PI. This API will be deprecated in the future releases. See SPARK-28264 for more details.
  warnings.warn(
```

```
Wall time: 19.8 s
```

In [19]:

```
%%time
model_results_by_id3[['recall', 'precision', 'accuracy', 'auc']] = model_results_by_id3[['recall', 'precision', 'accuracy', 'auc']].round(3)
model_results_by_id3.sort_values(by = 'id').reset_index(drop = True).head()
```

```
Wall time: 3.96 ms
```

Out[19]:

	id	recall	precision	accuracy	auc
0	1	0.644	0.959	0.923	0.852
1	2	0.714	0.968	0.943	0.877
2	3	0.682	0.966	0.925	0.867
3	4	0.706	0.958	0.931	0.877

In [ ]: