ML-RANDOMFORESTREGRESSION-CIHANERSOY

April 18, 2020

[1]:
```python
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.metrics import mean_squared_error, accuracy_score
from sklearn.preprocessing import LabelEncoder
from scipy.stats import randint
```

Necessary libraries are downloaded. Pandas and numpy are standard data science libraries. Sklearn is for prediction and hyperparameter tuning. Scipy library is used to generate random numbers for RandomizedSearchCV.

[2]:
```python
df = pd.read_csv('DelayedFlights.csv')
```

DelayedFlights.csv dataset is brought into a pandas dataframe.

[3]:
```python
df.describe()
```

[3]:
```
         Unnamed: 0       Year         Month    DayofMonth     DayOfWeek  \
count  1.936758e+06  1936758.0  1.936758e+06  1.936758e+06  1.936758e+06
mean   3.341651e+06     2008.0  6.111106e+00  1.575347e+01  3.984827e+00
std    2.066065e+06        0.0  3.482546e+00  8.776272e+00  1.995966e+00
min    0.000000e+00     2008.0  1.000000e+00  1.000000e+00  1.000000e+00
25%    1.517452e+06     2008.0  3.000000e+00  8.000000e+00  2.000000e+00
50%    3.242558e+06     2008.0  6.000000e+00  1.600000e+01  4.000000e+00
75%    4.972467e+06     2008.0  9.000000e+00  2.300000e+01  6.000000e+00
max    7.009727e+06     2008.0  1.200000e+01  3.100000e+01  7.000000e+00

            DepTime    CRSDepTime       ArrTime    CRSArrTime     FlightNum  \
count  1.936758e+06  1.936758e+06  1.929648e+06  1.936758e+06  1.936758e+06
mean   1.518534e+03  1.467473e+03  1.610141e+03  1.634225e+03  2.184263e+03
std    4.504853e+02  4.247668e+02  5.481781e+02  4.646347e+02  1.944702e+03
min    1.000000e+00  0.000000e+00  1.000000e+00  0.000000e+00  1.000000e+00
25%    1.203000e+03  1.135000e+03  1.316000e+03  1.325000e+03  6.100000e+02
50%    1.545000e+03  1.510000e+03  1.715000e+03  1.705000e+03  1.543000e+03
75%    1.900000e+03  1.815000e+03  2.030000e+03  2.014000e+03  3.422000e+03
max    2.400000e+03  2.359000e+03  2.400000e+03  2.400000e+03  9.742000e+03
```

```
       …      Distance        TaxiIn       TaxiOut     Cancelled  \
count  …  1.936758e+06  1.929648e+06  1.936303e+06  1.936758e+06
mean   …  7.656862e+02  6.812975e+00  1.823220e+01  3.268348e-04
std    …  5.744797e+02  5.273595e+00  1.433853e+01  1.807562e-02
min    …  1.100000e+01  0.000000e+00  0.000000e+00  0.000000e+00
25%    …  3.380000e+02  4.000000e+00  1.000000e+01  0.000000e+00
50%    …  6.060000e+02  6.000000e+00  1.400000e+01  0.000000e+00
75%    …  9.980000e+02  8.000000e+00  2.100000e+01  0.000000e+00
max    …  4.962000e+03  2.400000e+02  4.220000e+02  1.000000e+00

           Diverted  CarrierDelay  WeatherDelay      NASDelay  SecurityDelay  \
count  1.936758e+06  1.247488e+06  1.247488e+06  1.247488e+06   1.247488e+06
mean   4.003598e-03  1.917940e+01  3.703571e+00  1.502164e+01   9.013714e-02
std    6.314722e-02  4.354621e+01  2.149290e+01  3.383305e+01   2.022714e+00
min    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00   0.000000e+00
25%    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00   0.000000e+00
50%    0.000000e+00  2.000000e+00  0.000000e+00  2.000000e+00   0.000000e+00
75%    0.000000e+00  2.100000e+01  0.000000e+00  1.500000e+01   0.000000e+00
max    1.000000e+00  2.436000e+03  1.352000e+03  1.357000e+03   3.920000e+02

       LateAircraftDelay
count       1.247488e+06
mean        2.529647e+01
std         4.205486e+01
min         0.000000e+00
25%         0.000000e+00
50%         8.000000e+00
75%         3.300000e+01
max         1.316000e+03

[8 rows x 25 columns]
```

Exploratory data analysis. Describe method brings us count, mean, median vs. values of the dataset.

[4]:
```python
df.shape
```

[4]:
```
(1936758, 30)
```

Here is the number of rows and columns.

[5]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1936758 entries, 0 to 1936757
Data columns (total 30 columns):
Unnamed: 0           int64
Year                 int64
```

```
Month                 int64
DayofMonth            int64
DayOfWeek             int64
DepTime               float64
CRSDepTime            int64
ArrTime               float64
CRSArrTime            int64
UniqueCarrier         object
FlightNum             int64
TailNum               object
ActualElapsedTime     float64
CRSElapsedTime        float64
AirTime               float64
ArrDelay              float64
DepDelay              float64
Origin                object
Dest                  object
Distance              int64
TaxiIn                float64
TaxiOut               float64
Cancelled             int64
CancellationCode      object
Diverted              int64
CarrierDelay          float64
WeatherDelay          float64
NASDelay              float64
SecurityDelay         float64
LateAircraftDelay     float64
dtypes: float64(14), int64(11), object(5)
memory usage: 443.3+ MB
```

Exploratory data analysis. Different data types in the dataset by columns.

[6]:
```
Le_UniqueCarrier=LabelEncoder()
Le_Origin=LabelEncoder()
Le_Dest=LabelEncoder()
Le_CancellationCode=LabelEncoder()
```

We have already detected the object type columns and now we convert those object-type columns into numerical values. Since machine learning algorithms cannot deal with strings.

[7]:
```
df['UniqueCarrier_n'] = Le_UniqueCarrier.fit_transform(df['UniqueCarrier'])
df['Origin_n'] = Le_Origin.fit_transform(df['Origin'])
df['Dest_n'] = Le_Dest.fit_transform(df['Dest'])
df['CancellationCode_n'] = Le_CancellationCode.
 ↪fit_transform(df['CancellationCode'])
```

Object-type columns are converted into numerical type columns. New numerical columns are inserted into our dataset.

[8]:
```python
df.shape
```

[8]: (1936758, 34)

4 brand new columns are added.

[9]:
```python
df.drop(['UniqueCarrier', 'Origin', 'Dest', 'CancellationCode', 'TailNum',␣
↪'Unnamed: 0'], axis = 1, inplace=True)
```

4 Object-type columns, 1 extra index column and 1 other irrelevant (tail number) column are dropped.

[10]:
```python
df.shape
```

[10]: (1936758, 28)

The number of columns and rows are checked to see if adding and dropping are successful. So far so good.

[11]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1936758 entries, 0 to 1936757
Data columns (total 28 columns):
Year                   int64
Month                  int64
DayofMonth             int64
DayOfWeek              int64
DepTime                float64
CRSDepTime             int64
ArrTime                float64
CRSArrTime             int64
FlightNum              int64
ActualElapsedTime      float64
CRSElapsedTime         float64
AirTime                float64
ArrDelay               float64
DepDelay               float64
Distance               int64
TaxiIn                 float64
TaxiOut                float64
Cancelled              int64
Diverted               int64
CarrierDelay           float64
WeatherDelay           float64
NASDelay               float64
SecurityDelay          float64
LateAircraftDelay      float64
UniqueCarrier_n        int64
```

```
Origin_n               int64
Dest_n                 int64
CancellationCode_n     int64
dtypes: float64(14), int64(14)
memory usage: 413.7 MB
```

The data type of each column is checked.

[12]:
```python
df.isnull().sum()
```

[12]:
```
Year                       0
Month                      0
DayofMonth                 0
DayOfWeek                  0
DepTime                    0
CRSDepTime                 0
ArrTime                 7110
CRSArrTime                 0
FlightNum                  0
ActualElapsedTime       8387
CRSElapsedTime           198
AirTime                 8387
ArrDelay                8387
DepDelay                   0
Distance                   0
TaxiIn                  7110
TaxiOut                  455
Cancelled                  0
Diverted                   0
CarrierDelay          689270
WeatherDelay          689270
NASDelay              689270
SecurityDelay         689270
LateAircraftDelay     689270
UniqueCarrier_n            0
Origin_n                   0
Dest_n                     0
CancellationCode_n         0
dtype: int64
```

Null values are checked.

[13]:
```python
df.dropna(axis=0, how='any', inplace=True)
```

Rows that have null values are dropped.

[14]:
```python
df.isnull().sum()
```

[14]:
```
Year                   0
Month                  0
DayofMonth             0
DayOfWeek              0
DepTime                0
CRSDepTime             0
ArrTime                0
CRSArrTime             0
FlightNum              0
ActualElapsedTime      0
CRSElapsedTime         0
AirTime                0
ArrDelay               0
DepDelay               0
Distance               0
TaxiIn                 0
TaxiOut                0
Cancelled              0
Diverted               0
CarrierDelay           0
WeatherDelay           0
NASDelay               0
SecurityDelay          0
LateAircraftDelay      0
UniqueCarrier_n        0
Origin_n               0
Dest_n                 0
CancellationCode_n     0
dtype: int64
```

No more null values!

[15]:
```
df.reset_index(drop=True, inplace=True)
```

We reset index to tidy up our data set.

[16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1247488 entries, 0 to 1247487
Data columns (total 28 columns):
Year                 1247488 non-null int64
Month                1247488 non-null int64
DayofMonth           1247488 non-null int64
DayOfWeek            1247488 non-null int64
DepTime              1247488 non-null float64
CRSDepTime           1247488 non-null int64
ArrTime              1247488 non-null float64
```

```
CRSArrTime            1247488 non-null int64
FlightNum             1247488 non-null int64
ActualElapsedTime     1247488 non-null float64
CRSElapsedTime        1247488 non-null float64
AirTime               1247488 non-null float64
ArrDelay              1247488 non-null float64
DepDelay              1247488 non-null float64
Distance              1247488 non-null int64
TaxiIn                1247488 non-null float64
TaxiOut               1247488 non-null float64
Cancelled             1247488 non-null int64
Diverted              1247488 non-null int64
CarrierDelay          1247488 non-null float64
WeatherDelay          1247488 non-null float64
NASDelay              1247488 non-null float64
SecurityDelay         1247488 non-null float64
LateAircraftDelay     1247488 non-null float64
UniqueCarrier_n       1247488 non-null int64
Origin_n              1247488 non-null int64
Dest_n                1247488 non-null int64
CancellationCode_n    1247488 non-null int64
dtypes: float64(14), int64(14)
memory usage: 266.5 MB
```

Dataset is controlled for last time and now we are ready to train our model.

[17]:
```
y=df.loc[:, 'DepDelay']
```

Target variable is set.

[18]:
```
X=df.loc[:, df.columns != 'DepDelay']
```

The rest of the columns will be used as predictors.

[19]:
```
X_train, X_test, y_train, y_test= train_test_split(X,y, test_size=0.2,␣
 ↪random_state=12)
```

Train and test data are splitted. random_state is used to get same result if code is implemented later again.

[20]:
```
param_grid={"n_estimators":randint(1,9),
             "max_depth": (2,7),
            "max_features": randint(1,9),
            "min_samples_leaf": randint(1,9)}
```

Hyperparameter tuning method is decided to choose parameters such as max_depth, n_estimators, max_features, and min_sample_leaf.

[21]:
```
RFReg=RandomForestRegressor(n_jobs=-1, random_state=42)
```

RandomForestRegressor method is used. n_job=-1 helps computer to use full capacity CPU. random_state is again for later implementations of the same code.

[22]:
```
RFReg_cv= RandomizedSearchCV(RFReg, param_grid, cv=5)
```

I decided to use RandomizedSearchCV rather than plain hyper parameter tuning (GridSearchCV) to make parameter choosing process shorter.

[23]:
```
RFReg_cv.fit(X_train, y_train)
```

[23]:
```
RandomizedSearchCV(cv=5, error_score='raise-deprecating',
                   estimator=RandomForestRegressor(bootstrap=True,
                                                   criterion='mse',
                                                   max_depth=None,
                                                   max_features='auto',
                                                   max_leaf_nodes=None,
                                                   min_impurity_decrease=0.0,
                                                   min_impurity_split=None,
                                                   min_samples_leaf=1,
                                                   min_samples_split=2,
                                                   min_weight_fraction_leaf=0.0,
                                                   n_estimators='warn',
                                                   n_jobs=-1, oob_score=False,
                                                   random_state…
                                        'max_features':
<scipy.stats._distn_infrastructure.rv_frozen object at 0x7f73676bbfd0>,
                                        'min_samples_leaf':
<scipy.stats._distn_infrastructure.rv_frozen object at 0x7f73669baed0>,
                                        'n_estimators':
<scipy.stats._distn_infrastructure.rv_frozen object at 0x7f73669ba5d0>},
                   pre_dispatch='2*n_jobs', random_state=None, refit=True,
                   return_train_score=False, scoring=None, verbose=0)
```

Training is implemented.

[24]:
```
RFReg_cv.best_params_
```

[24]:
```
{'max_depth': 7, 'max_features': 6, 'min_samples_leaf': 7, 'n_estimators': 5}
```

These parameters are chosen by the algorithm as the most optimized ones.

[25]:
```
y_pred=RFReg_cv.predict(X_test)
```

Prediction is implemented on the test data.

[26]:
```
mean_squared_error(y_test, y_pred)
```

[26]:
```
337.56074148431287
```

Calculation of MSE.

[27]:
```
rmse=mean_squared_error(y_test, y_pred)**0.5
```

Calculation of avarage value error.

[28]:
```
rmse
```

[28]: 18.372826170306865

It is pretty close to the value that I found by implementing linear regression, yet slightly worse than that.

[29]:
```
RFReg_cv.score(X_test, y_test)
```

[29]: 0.9052527483794639

Calculation of R2

The Theory Behind My Solution

1) In first assignment, I used a feature selection method to select most relevant features. However, in this assignment I didn't use feature selection. Because second assignment is still a regression problem, and although there are lots of different feature selection algorithms, same feature selection algorithm would apply to this problem. The reason is that both input and output data are numerical. So, I wanted to see the difference how feature selection algorithm affects the result as well as random forest method.

2) I converted string type columns into numeric type. Because ML algorithms cannot handle string type data. In the first assignment I dropped string type columns thinking that they are irrelevant, as a matter of fact, I was supposed to leave this decision(column eliminating decision) to feature selection algorithm. So, this is expected to improve the result of a model.

3) Some resources indicate that we do not need to drop null values in decision tree/random forest model because the process itself already handles it, yet I did drop null values.

4) The biggest factor that may differ my model from others is hyper parameter tuning. In decision tree/random forest method, we need to detect the depth of each tree, the number of trees and the number of leaf in each tree vs. If we (rather than algorithm) decide all these details, the power of model will be limited. However, if we leave this decision to an unsupervised algorithm(hyper parameter tuning algortihm) then algorithm will find the most optimized parameters of algorithm on its own. Here we have two options: One is gridsearch which try each and every single parameter in the algorithm. This one takes way too long to process. The other one is randomized search. This one takes relatively shorter because it doesnt try every single parameter. It chooses a few parameters randomly and trys to find most optimized ones from randomly selected parameters. I used the latter as the computation capacity of my Personal Computer is limited.

5) I want to mention "n_jobs=-1" more particularly which does pretty good job to use whole computational capacity of CPU, thanks to which it takes shorter to train your data.

6)The result : R2 score is slightly worse than my previous model which is linear regression. (90.5% vs 92.6%) The reason might be feature selection algorithm or even maybe random_state number. As this is not a classification problem evaulation method cannot be confusion matrix. RMSE, MSE and accuracy score are only ways to measure the performance of our model.