```
from google.colab import files
uploaded = files.upload()
```

选择文件 2 个文件

- **arr_big_data.csv**(application/vnd.ms-excel) - 141518235 bytes, last modified: 2020/4/30 - 100% done
- **dep_big_data.csv**(application/vnd.ms-excel) - 139548527 bytes, last modified: 2020/4/30 - 100% done

```
Saving arr_big_data.csv to arr_big_data.csv
Saving dep_big_data.csv to dep_big_data.csv
```

```
import pandas as pd
data = pd.read_csv('arr_big_data.csv')
pd.DataFrame.from_records(data)
# del data['Unnamed: 0']
# data
```

.	dest_RIC	dest_RNO	dest_ROA	dest_ROC	dest_RSW	dest_SAN	dest_SAT	dest_SAV	dest_SBA	de
…	0	0	0	0	0	0	0	0	0	
…	0	0	0	0	0	0	0	0	0	
…	0	0	0	0	0	0	0	0	0	
…	0	0	0	0	0	0	0	0	0	
…	0	0	0	0	1	0	0	0	0	
…	…	…	…	…	…	…	…	…	…	…
…	0	0	0	0	0	0	0	0	0	
…	0	0	0	0	0	0	0	0	0	
…	0	0	0	0	0	0	0	0	0	
…	0	0	0	0	0	0	0	0	0	
…	0	0	0	0	0	0	0	0	0	

Predictions for Arrival Delay

```
import pandas as pd
from sklearn.model_selection import train_test_split
import xgboost as xgb
from sklearn.metrics import mean_squared_error
import numpy as np

# data.dropna(axis=0, subset=['total_arr_delay'], inplace=True)

y = data.total_arr_delay
X = data.drop(['total_arr_delay'], axis=1).select_dtypes(exclude=['object'])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=888)
```

Mean Absolute Error results

```
data_dmatrix = xgb.DMatrix(data=X,label=y)
# dtrain = xgb.DMatrix(X_train, label=y_train)
# dtest = xgb.DMatrix(X_test, label=y_test)

xg_reg = xgb.XGBRegressor(objective ='reg:linear', colsample_bytree = 0.3, learning_rate = 0.1,
                          max_depth = 5, alpha = 10, n_estimators = 10)

xg_reg.fit(X_train,y_train)

preds = xg_reg.predict(X_test)

rmse = np.sqrt(mean_squared_error(y_test, preds))
print("RMSE: %f" % (rmse))
```

```
RMSE: 23.745028
```

Using k-fold Cross Validation for model tuning

```
# params = {"objective":"reg:linear",'colsample_bytree': 0.3,'learning_rate': 0.1,
#                 'max_depth': 5, 'alpha': 10}

# cv_results = xgb.cv(dtrain=data_dmatrix, params=params, nfold=3,
#                     num_boost_round=50,early_stopping_rounds=10,metrics="rmse", as_panc

params = {"objective":'reg:squarederror','colsample_bytree': 0.3,'learning_rate': 0.35,
                'max_depth': 7, 'alpha': 10, 'subsample': 0.7, 'gamma': 0.1}
cv_results_g1 = xgb.cv(dtrain=data_dmatrix, params=params, nfold=5,
                       num_boost_round=200,early_stopping_rounds=10,metrics="rmse", as_pandas

cv_results_g1.head()
```

	train-rmse-mean	train-rmse-std	test-rmse-mean	test-rmse-std
0	29.512923	0.038624	29.551592	0.161476
1	23.782266	2.543996	23.860410	2.448174
2	22.744114	3.272639	22.890786	3.134779
3	21.819126	3.650699	22.018433	3.552576
4	21.699805	3.681405	21.923173	3.580048

Better RMSE

```
print((cv_results_g1["test-rmse-mean"]).tail(1))
```

```
199    13.856875
Name: test-rmse-mean, dtype: float64
```

```
xg_reg = xgb.train(params=params, dtrain=data_dmatrix, num_boost_round=10)
```

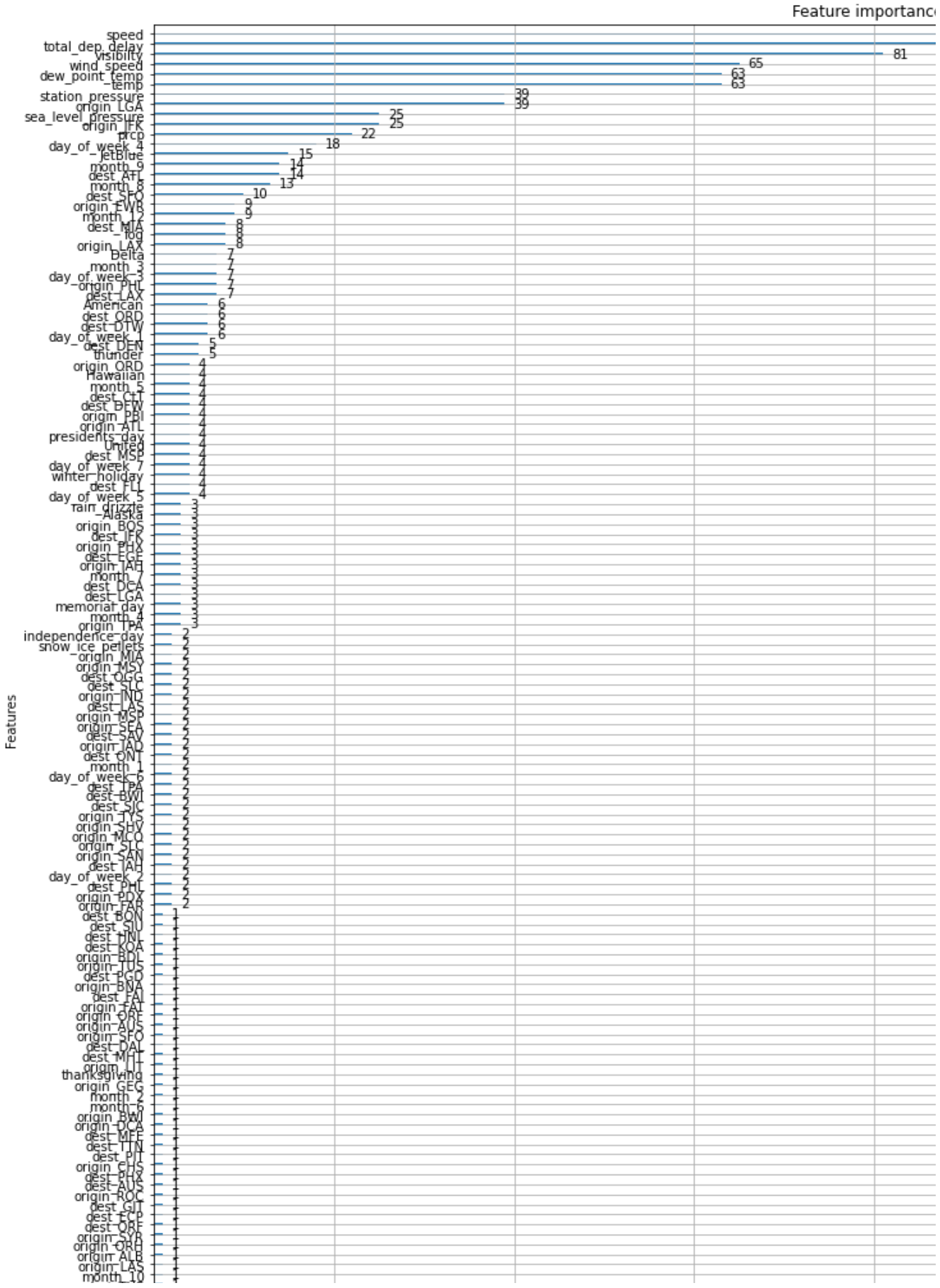
Visualize Boosting Trees and Feature Importance

```
from  numpy  import  loadtxt
from  xgboost  import  XGBClassifier
from  xgboost  import  plot_tree
import  matplotlib.pyplot  as  plt

plt.rcParams["figure.figsize"]  =  20,20
plot_tree(xg_reg,  num_trees=0,  rankdir='LR')
#  xgb.plot_tree(xg_reg,num_trees=0)
#  xgb.plot_tree(xg_reg,  num_trees=0,  rankdir='LR')
#  plt.show()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1bb7c230b8>
```

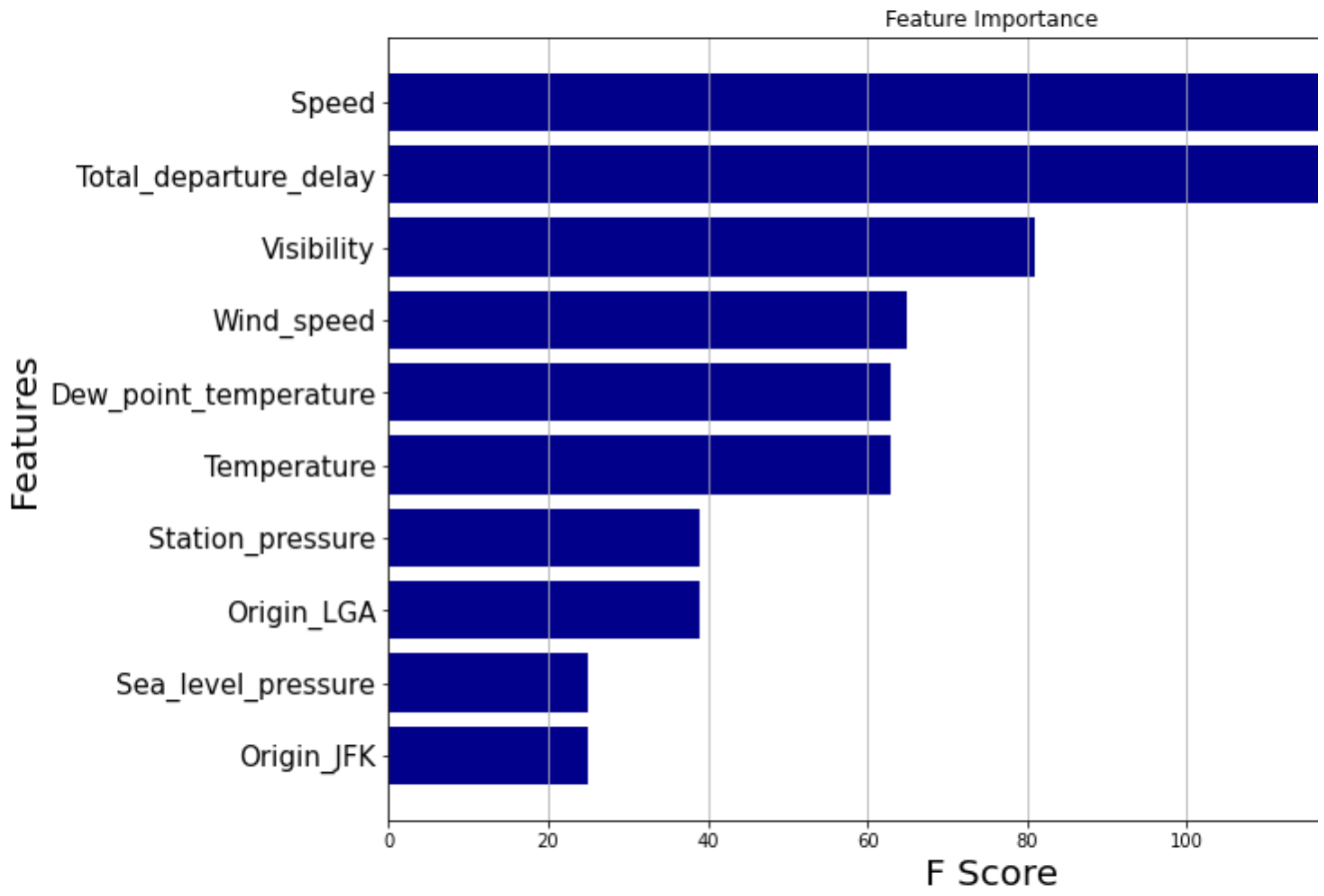
Top Features

```
xgb.plot_importance(xg_reg)
plt.rcParams["figure.figsize"]  =  20,20
plt.show()
```




```
plt.figure(figsize=[12,8])

import matplotlib.pyplot as plt
from matplotlib import cm
import numpy as np

label = ['Speed','Total_departure_delay','Visibility','Wind_speed','Dew_point_temperature','Temperature','Sta
```

```
x = [144,127,81,65,63,63,39,39,25,25]

idx = np.arange(len(x))
# color = cm.jet(np.array(x)/max(x))
plt.barh(idx, x, color='darkblue')
plt.yticks(idx,label,fontsize=15)
plt.grid(axis='x')
plt.xlabel('F Score',fontsize=20)
plt.ylabel('Features',fontsize=20)
plt.title('Feature Importance')
plt.gca().invert_yaxis()

plt.show()
```



Predictions for Departure Delay

```
import pandas as pd
data = pd.read_csv('dep_big_data.csv')
pd.DataFrame.from_records(data)
```

tation_pressure	visibilty	wind_speed	prcp	fog	rain_drizzle	snow_ice_pellets	hail	thunde
1010.5	10.0	8.6	0.62	0	0	0	0	
1018.3	6.9	3.9	0.00	1	0	0	0	
938.9	10.0	6.2	0.00	0	0	0	0	
982.4	10.0	5.7	0.00	0	0	0	0	
1023.6	10.0	6.1	0.00	0	0	0	0	
...	...	...	...	...	...	...	...	
989.3	10.0	5.3	0.00	0	0	0	0	
977.0	5.2	11.0	0.66	1	0	1	0	
0.0	10.0	10.7	0.00	0	0	0	0	

```
import pandas as pd
from sklearn.model_selection import train_test_split
import xgboost as xgb
from sklearn.metrics import mean_squared_error
import numpy as np

# data.dropna(axis=0, subset=['total_dep_delay'], inplace=True)

y = data.total_dep_delay
X = data.drop(['total_dep_delay'], axis=1).select_dtypes(exclude=['object'])


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=123)
```

Mean Absolute Error results

```
data_dmatrix = xgb.DMatrix(data=X,label=y)

xg_reg = xgb.XGBRegressor(objective ='reg:linear', colsample_bytree = 0.3, learning_rate = 0.1,
                          max_depth = 5, alpha = 10, n_estimators = 10)


xg_reg.fit(X_train,y_train)

preds = xg_reg.predict(X_test)


rmse = np.sqrt(mean_squared_error(y_test, preds))
print("RMSE: %f" % (rmse))
```

```
RMSE: 27.676985
```

Using k-fold Cross Validation for model tuning

```
# params = {"objective":"reg:linear",'colsample_bytree': 0.3,'learning_rate': 0.1,
#                                 'max_depth': 5, 'alpha': 10}
```

```
# cv_results = xgb.cv(dtrain=data_dmatrix, params=params, nfold=3,
#                                         num_boost_round=50,early_stopping_rounds=10,metrics="rmse", as_panc

params = {"objective":'reg:squarederror','colsample_bytree': 0.3,'learning_rate': 0.1,
                                'max_depth': 7, 'alpha': 10, 'subsample': 0.7, 'gamma': 0.1}
cv_results_g1 = xgb.cv(dtrain=data_dmatrix, params=params, nfold=5,
                                        num_boost_round=200,early_stopping_rounds=10,metrics="rmse", as_pandas
```

```
cv_results_g1.head()
```

	train-rmse-mean	train-rmse-std	test-rmse-mean	test-rmse-std
0	33.669781	0.044570	33.679391	0.188015
1	32.377024	0.045918	32.402808	0.187011
2	31.299827	0.045561	31.335240	0.183580
3	30.384858	0.045082	30.436723	0.186172
4	29.630687	0.047559	29.693999	0.186524

Better RMSE

```
print((cv_results_g1["test-rmse-mean"]).tail(1))
```

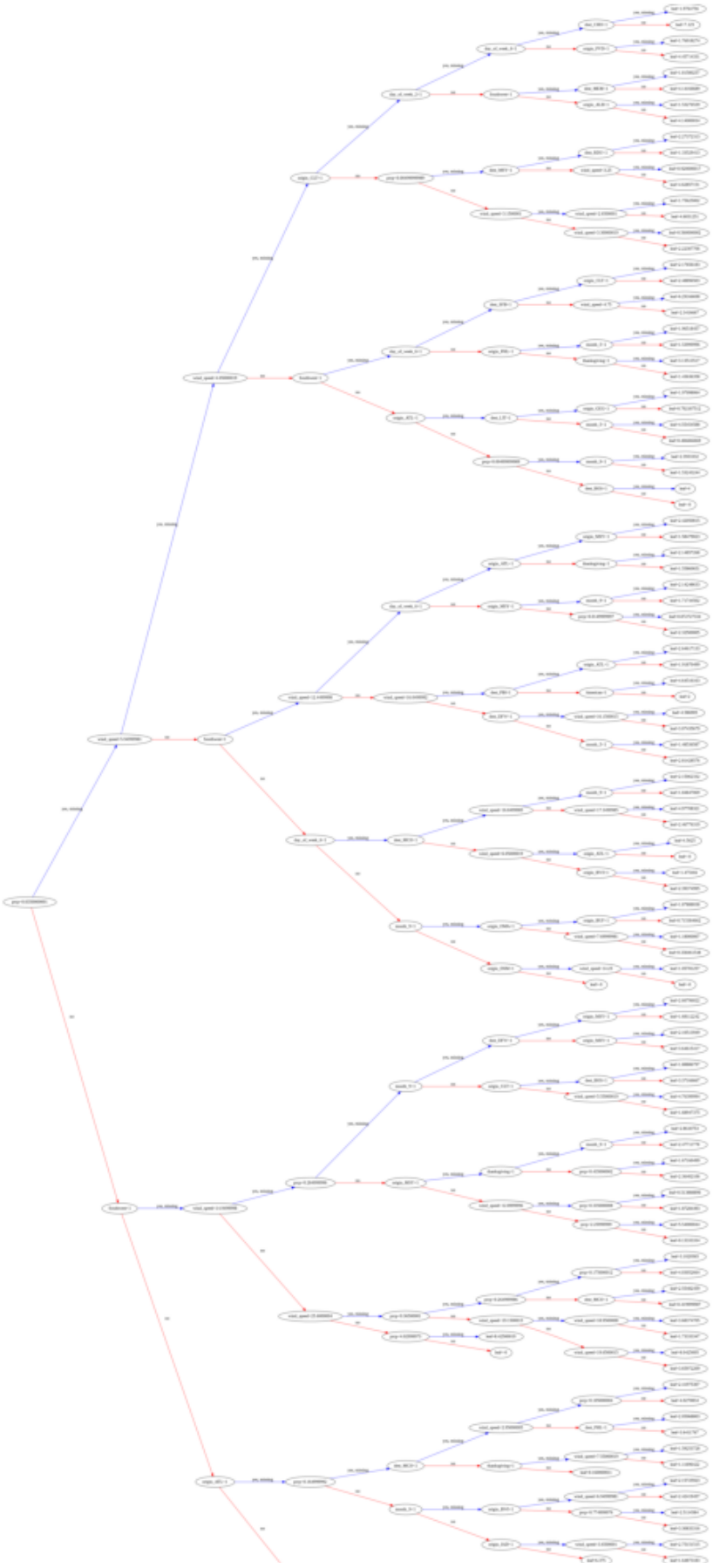
```
199    25.592875
Name: test-rmse-mean, dtype: float64
```

```
xg_reg = xgb.train(params=params, dtrain=data_dmatrix, num_boost_round=10)
```

Visualize Boosting Trees and Feature Importance

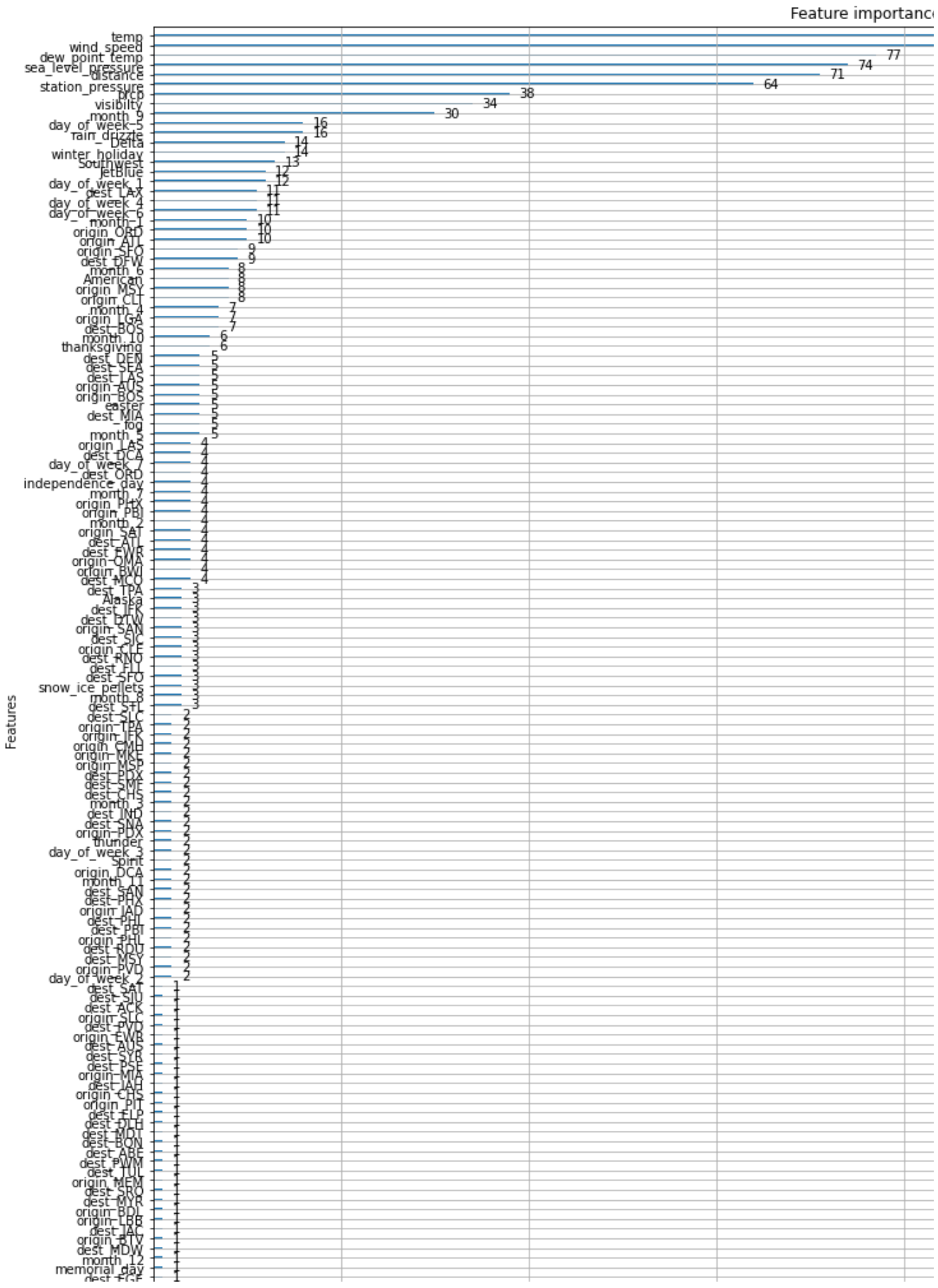
```
from numpy import loadtxt
from xgboost import XGBClassifier
from xgboost import plot_tree
import matplotlib.pyplot as plt

plt.rcParams["figure.figsize"] = 20,20
plot_tree(xg_reg, num_trees=0, rankdir='LR')
# xgb.plot_tree(xg_reg,num_trees=0)
# xgb.plot_tree(xg_reg, num_trees=0, rankdir='LR')
# plt.show()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1bb5dec588>
```



Top Features

```
xgb.plot_importance(xg_reg)
plt.rcParams['figure.figsize'] = [5, 5]
plt.show()
```




```
plt.figure(figsize=[12,8])

import matplotlib.pyplot as plt
from matplotlib import cm
import numpy as np
```

```
label  =  ['Temperature','Wind_speed','Dew_point_temperature','Sea_level_pressure','Distance','Station_pressure
x  =    [138,109,77,74,71,64,38,34,30,16,16]

idx  =  np.arange(len(x))
#  color  =  cm.jet(np.array(x)/max(x))
plt.barh(idx,  x,  color='darkblue')
plt.yticks(idx,label,fontsize=15,horizontalalignment="right")
plt.grid(axis='x')
plt.xlabel('F  Score',fontsize=20)
plt.ylabel('Features',fontsize=20)
plt.title('Feature  Importance')
plt.gca().invert_yaxis()
plt.show()
```