

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



选择文件 big_data.csv

- **big_data.csv**(application/vnd.ms-excel) - 139769656 bytes, last modified: 2020/4/28 - 100% done
- Saving big_data.csv to big_data.csv

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



	arr_delay	distance	presidents_day	easter	memorial_day	independence_day	labor_d
0	-7	270	0	0	0	0	
1	5	1995	0	0	0	0	
2	7	621	0	0	0	0	
3	-33	2065	0	0	0	0	
4	-19	1771	0	0	0	0	
...
177507	8	337	0	0	0	0	
177508	10	1024	0	1	0	0	
177509	44	1585	0	1	0	0	
177510	-15	337	0	0	0	0	
177511	-10	624	0	0	0	0	

177512 rows × 382 columns

▼ 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=['arr_delay'], inplace=True)
```

```
y = data.arr_delay
X = data.drop(['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=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)

[18:01:39] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror

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

RMSE: 31.686198

```

▼ 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_pandas=True)

[18:01:56] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror
[18:02:01] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror
[18:02:04] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror

cv_results.head()

```

```

train-rmse-mean  train-rmse-std  test-rmse-mean  test-rmse-std
0      41.508718      0.103943      41.521145      0.207252
1      39.282958      1.427370      39.298312      1.723997
2      38.255065      2.349752      38.284343      2.695878
3      38.171123      2.361523      38.239312      2.694978
4      37.278604      3.411559      37.357897      3.752378

```

▼ Better RMSE

```

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

49      19.407878
Name: test-rmse-mean, dtype: float64

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

[18:04:26] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror

```

▼ Visualizations

```

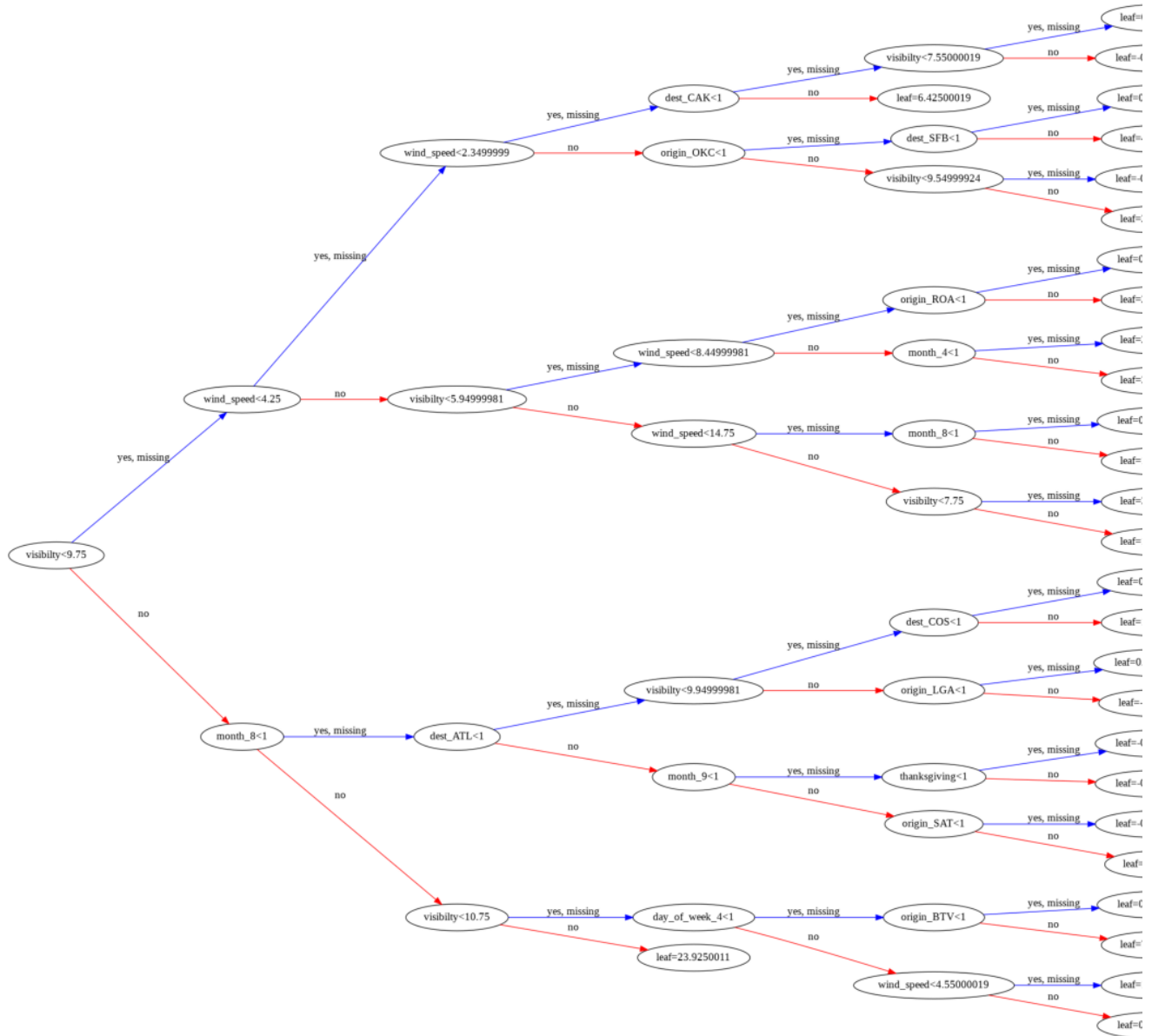
from numpy import loadtxt
from xgboost import XGBClassifier
from xgboost import plot_tree

```

```
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()
```

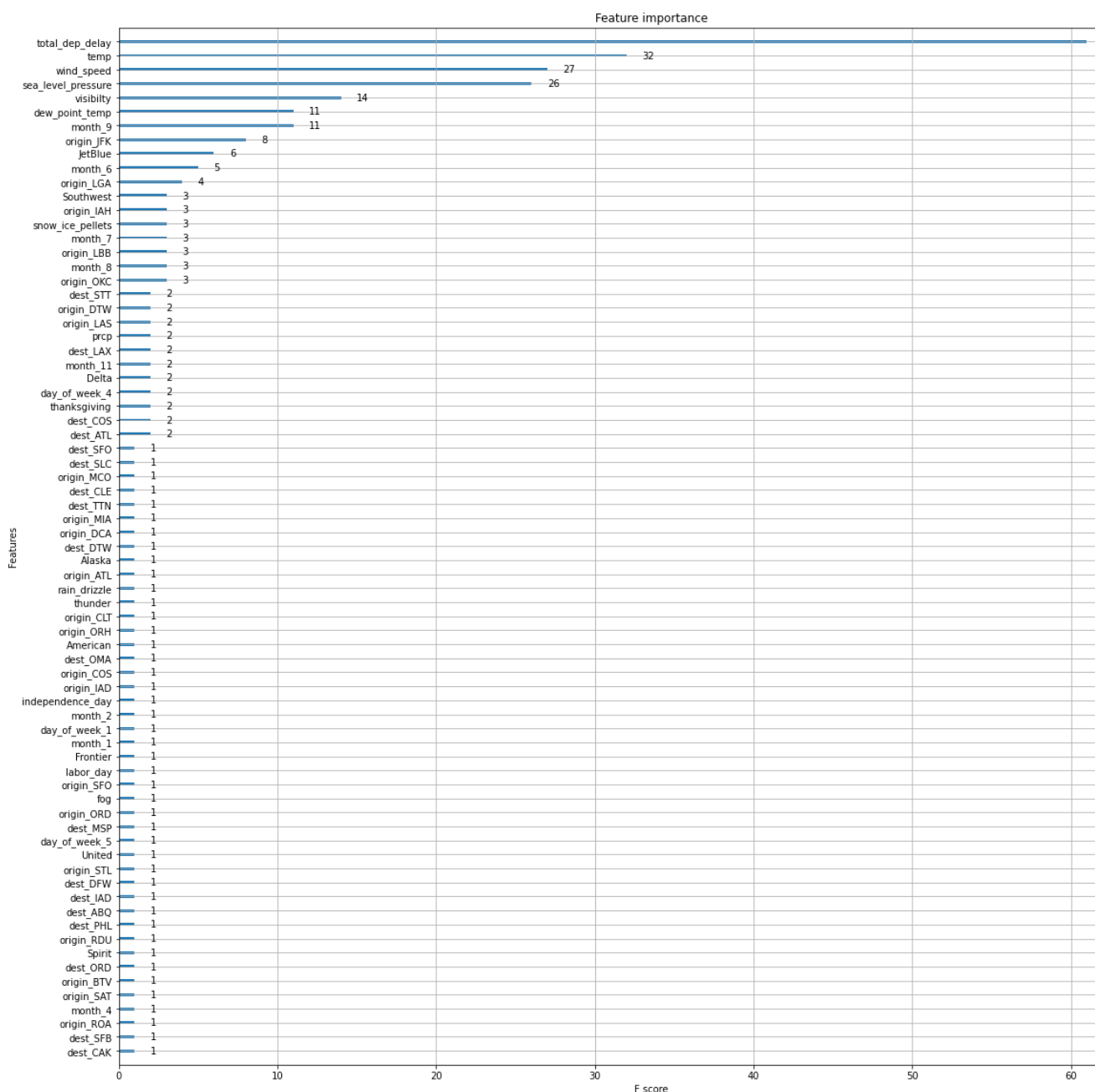
↳ <matplotlib.axes._subplots.AxesSubplot at 0x7f0c30acaeb8>



▼ Top Features

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





▼ Predictions for Departure Delay

```
import pandas as pd
data = pd.read_csv('big_data.csv')
```

```
pd.DataFrame.from_records(data)
```

	arr_delay	distance	presidents_day	easter	memorial_day	independence_day	labor_d
0	-7	270	0	0	0	0	
1	5	1995	0	0	0	0	
2	7	621	0	0	0	0	
3	-33	2065	0	0	0	0	
4	-19	1771	0	0	0	0	
...
177507	8	337	0	0	0	0	
177508	10	1024	0	1	0	0	
177509	44	1585	0	1	0	0	
177510	-15	337	0	0	0	0	
177511	-10	624	0	0	0	0	

177512 rows × 382 columns

```
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)

[19:02:35] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror

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

RMSE: 32.142434

▼ 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_pandas=True)
```

```
[19:04:11] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror
[19:04:15] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror
[19:04:19] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror
```

```
cv_results.head()
```

```

train-rmse-mean  train-rmse-std  test-rmse-mean  test-rmse-std
0      38.656838      0.041707      38.663035      0.085181
1      37.358302      0.049064      37.371316      0.079493
2      36.254335      0.051466      36.275592      0.077880
3      35.338801      0.054763      35.367652      0.076687
4      34.576529      0.050749      34.611799      0.078184
```

▼ Better RMSE

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

```

49      11.025499
Name: test-rmse-mean, dtype: float64
```

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

```
[19:07:16] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror
```

▼ Visualizations

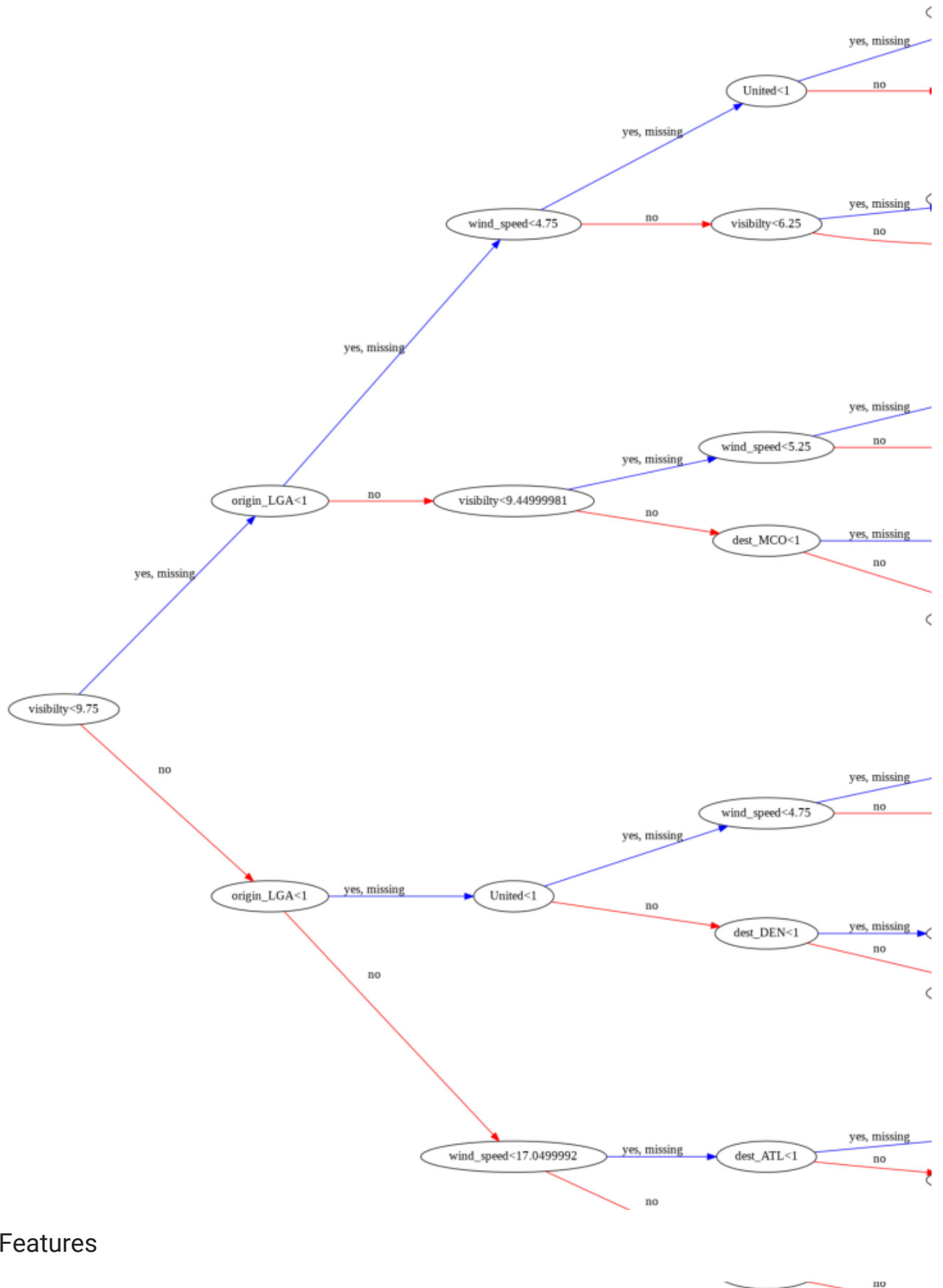
```
from numpy import loadtxt
from xgboost import XGBClassifier
from xgboost import plot_tree
import matplotlib.pyplot as plt
```

```
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 0x7f0c30aca208>



▼ Top Features

```

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

```