Flight Status 2018 Dicision Tree

In [1]:
```
# import the neede package
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
from sklearn.metrics import confusion_matrix
from datetime import date
import calendar
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
warnings.warn('DelftStack')
warnings.warn('Do not show this message')
print('No Warning Shown')
```

```
No Warning Shown
```

In [3]:
```
flight = pd.read_csv("/Users/UE/Desktop/flight2018.csv")
```

In [4]:
```
flight = flight[['Month','DayOfWeek','Distance','Origin','Airline','DepTimeBlk','DepDelay']]
flight
```

Out[4]:

	Month	DayOfWeek	Distance	Origin	Airline	DepTimeBlk	DepDelay
0	1	2	145.0	ABY	Endeavor Air Inc.	1200-1259	-5.0
1	1	3	145.0	ABY	Endeavor Air Inc.	1200-1259	-5.0
2	1	4	145.0	ABY	Endeavor Air Inc.	1200-1259	-9.0
3	1	5	145.0	ABY	Endeavor Air Inc.	1200-1259	-12.0
4	1	6	145.0	ABY	Endeavor Air Inc.	1400-1459	-5.0
...	...	...	...	...	...	...	...
5689507	9	2	133.0	SCE	Air Wisconsin Airlines Corp	1400-1459	-12.0
5689508	9	2	239.0	IAD	Air Wisconsin Airlines Corp	1200-1259	-11.0
5689509	9	2	272.0	EVV	Air Wisconsin Airlines Corp	1000-1059	-14.0
5689510	9	2	738.0	ORD	Air Wisconsin Airlines Corp	1400-1459	-7.0
5689511	9	2	738.0	HPN	Air Wisconsin Airlines Corp	1800-1859	-6.0

5689512 rows × 7 columns

In [5]:
```
flight['Month'] = flight['Month'].apply(lambda x: calendar.month_abbr[x])
flight['Month']
```

Out[5]:
```
0          Jan
1          Jan
2          Jan
3          Jan
4          Jan
          ...
5689507    Sep
5689508    Sep
5689509    Sep
5689510    Sep
5689511    Sep
Name: Month, Length: 5689512, dtype: object
```

In [6]:
```
def Week_Eng(value):
    if value == "1":
        return "Mon"
    if value == "2":
        return "Tue"
    if value == "3":
        return "Wed"
    if value == "4":
        return "Thu"
    if value == "5":
        return "Fri"
    if value == "6":
        return "Sat"
    else:
        return "Sun"

flight['Week_Eng'] = flight['DayOfWeek'].map(Week_Eng)
display(flight.head())
```

	Month	DayOfWeek	Distance	Origin	Airline	DepTimeBlk	DepDelay	Week_Eng
0	Jan	2	145.0	ABY	Endeavor Air Inc.	1200-1259	-5.0	Sun
1	Jan	3	145.0	ABY	Endeavor Air Inc.	1200-1259	-5.0	Sun
2	Jan	4	145.0	ABY	Endeavor Air Inc.	1200-1259	-9.0	Sun
3	Jan	5	145.0	ABY	Endeavor Air Inc.	1200-1259	-12.0	Sun
4	Jan	6	145.0	ABY	Endeavor Air Inc.	1400-1459	-5.0	Sun

def Mon_Eng(value): if value == "1": return "Jan" if value == "2": return "Feb" if value == "3": return "Mar" if value == "4": return "Apr" if value == "5": return "May" if value == "6": return "Jun" if value == "7": return "Jul" if value == "8": return "Aug" if value == "9": return "Sep" if value == "10": return "Oct" if value == "11": return "Nov" else: return "Dec"

flight['Mon_Eng'] = flight['Month'].map(Mon_Eng) display(flight.head())

In [7]:
```
def Flight_model(value):
    if value <= 15.0:
        return "normal"
    else:
        return "abnormal"

flight['Flight_model'] = flight['DepDelay'].map(Flight_model)
display(flight.head())
```

	Month	DayOfWeek	Distance	Origin	Airline	DepTimeBlk	DepDelay	Week_Eng	Flight_model
0	Jan	2	145.0	ABY	Endeavor Air Inc.	1200-1259	-5.0	Sun	normal
1	Jan	3	145.0	ABY	Endeavor Air Inc.	1200-1259	-5.0	Sun	normal
2	Jan	4	145.0	ABY	Endeavor Air Inc.	1200-1259	-9.0	Sun	normal
3	Jan	5	145.0	ABY	Endeavor Air Inc.	1200-1259	-12.0	Sun	normal
4	Jan	6	145.0	ABY	Endeavor Air Inc.	1400-1459	-5.0	Sun	normal

def Dep_TB(value): if value == "0600-0659": return "6" if value == "0700-0759": return "7" if value == "0800-0859": return "8" if value == "0900-0959": return "9" if value == "1000-1059": return "10" if value == "1100-1159": return "11" if value == "1200-1259": return "12" if value == "1300-1359": return "13" if value == "1400-1459": return "14" if value == "1500-1559": return "15" if value == "1600-1659": return "16" if value == "1700-1759": return "17" if value == "1800-1859": return "18" if value == "1900-1959": return "19" if value == "2000-2059": return "20" if value == "2100-2159": return "21" if value == "2200-2259": return "22" if value == "2300-2359": return "23" else: return "0"

flight['Dep_TB'] = flight['DepTimeBlk'].map(Dep_TB) display(flight.head())

In [8]:
```
flight_train, flight_test = train_test_split( flight, test_size = 0.33, random_state = 7)
```

In [9]:
```
print('Original number of instances before partitioning: ', flight.shape[0],
      '\nNumber of instances in Training set: ', flight_train.shape[0],
      '\nNumber of instances in Test set: ', flight_test.shape[0])
```

```
Original number of instances before partitioning:  5689512
Number of instances in Training set:  3811973
Number of instances in Test set:  1877539
```

In [10]:
```
print('Proportion of training instances: ', flight_train.shape[0]/flight.shape[0]*100,
      '\nProportion of test instances: ', flight_test.shape[0]/flight.shape[0]*100)
```

```
Proportion of training instances:  66.99999929695201
Proportion of test instances:  33.000000703047995
```

In [12]:
```
X = flight_train[['Month', 'Week_Eng', 'Distance', 'Origin', 'Airline']]
```

In [13]:
```
Y = flight_train[['Flight_model']]
Y
```

Out[13]:

	Flight_model
482101	normal
5448067	normal
1692593	normal
2342318	normal
1135707	normal
...	...
3905091	normal
2632182	normal
2671129	abnormal
3335364	normal
585903	normal

3811973 rows × 1 columns

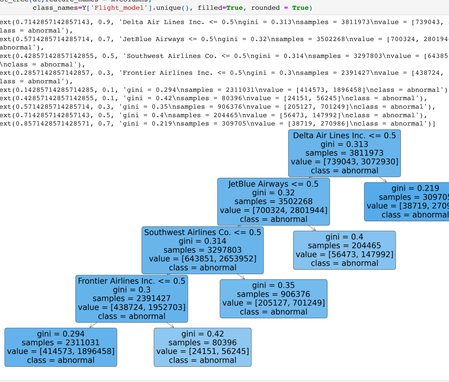
In [17]:
```
Org_dum = pd.get_dummies(X['Origin'])
AL_dum = pd.get_dummies(X['Airline'])
Mon_dum = pd.get_dummies(X['Month'])
Week_dum = pd.get_dummies(X['Week_Eng'])
```

In [18]:
```
X = pd.concat((X[['Distance']],Org_dum, AL_dum, Mon_dum,Week_dum), axis = 1)
```

In [19]:
```
dt = DecisionTreeClassifier(criterion='gini', max_leaf_nodes=5).fit(X, Y)
```

In [20]:
```
plt.figure(figsize=(40,20))
plot_tree(dt,feature_names = X.columns,
          class_names=Y['Flight_model'].unique(), filled=True, rounded = True)
```

Out[20]:
```
[Text(0.7142857142857143, 0.9, 'Delta Air Lines Inc. <= 0.5\ngini = 0.313\nsamples = 3811973\nvalue = [739043, 3072930]\
nclass = abnormal'),
 Text(0.5714285714285714, 0.7, 'JetBlue Airways <= 0.5\ngini = 0.32\nsamples = 3502268\nvalue = [700324, 2801944]\nclass
= abnormal'),
 Text(0.4285714285714285, 0.5, 'Southwest Airlines Co. <= 0.5\ngini = 0.314\nsamples = 3297803\nvalue = [643851, 265395
2]\nclass = abnormal'),
 Text(0.2857142857142857, 0.3, 'Frontier Airlines Inc. <= 0.5\ngini = 0.3\nsamples = 2391427\nvalue = [438724, 1952703]\
nclass = abnormal'),
 Text(0.1428571428571428, 0.1, 'gini = 0.294\nsamples = 2311031\nvalue = [414573, 1896458]\nclass = abnormal'),
 Text(0.4285714285714285, 0.1, 'gini = 0.42\nsamples = 80396\nvalue = [24151, 56245]\nclass = abnormal'),
 Text(0.5714285714285714, 0.3, 'gini = 0.35\nsamples = 906376\nvalue = [205127, 701249]\nclass = abnormal'),
 Text(0.7142857142857143, 0.5, 'gini = 0.4\nsamples = 204465\nvalue = [56473, 147992]\nclass = abnormal'),
 Text(0.8571428571428571, 0.7, 'gini = 0.219\nsamples = 309705\nvalue = [38719, 270986]\nclass = abnormal')]
```



In [22]:
```
X_test = flight_test[['Month', 'Week_Eng', 'Distance', 'Origin', 'Airline']]
Y_test = flight_test[['Flight_model']]
Org_dumt = pd.get_dummies(X_test['Origin'])
AL_dumt = pd.get_dummies(X_test['Airline'])
Mon_dumt = pd.get_dummies(X_test['Month'])
Week_dumt = pd.get_dummies(X_test['Week_Eng'])
X_test = pd.concat((X_test[['Distance']], Org_dumt, AL_dumt,Mon_dumt,Week_dumt), axis = 1)
```

In [23]:
```
pred = dt.predict(X_test)
pred
```

Out[23]:
```
array(['normal', 'normal', 'normal', ..., 'normal', 'normal', 'normal'],
      dtype=object)
```

In [24]:
```
cm = confusion_matrix(Y_test, pred)
cm
```

Out[24]:
```
array([[      0,  364766],
       [      0, 1512773]])
```

In [25]:
```
TN = cm[0][0]
FP = cm[0][1]
FN = cm[1][0]
TP = cm[1][1]

print('TN: ', TN,
      '\nFP: ', FP,
      '\nFN: ', FN,
      '\nTP: ', TP,)
```

```
TN:  0
FP:  364766
FN:  0
TP:  1512773
```

In [26]:
```
Accuracy = (TN+TP)/(TN+TP+FP+FN)
print('Accuracy: ' , Accuracy)
Error_Rate = 1 - Accuracy
print('Error Rate:', Error_Rate)
Sensitivity = Recall = TP / (TP + FN)
print("Sensitivity:", Sensitivity)
Specificity = TN/(TN + FP)
print('Specificity:', Specificity)
Precision = TP/(TP + FP)
print('Precision:', Precision)
F1 = 2 * (Precision * Recall)/(Precision + Recall)
print('F1:', F1)
F2 = 5 * (Precision * Recall) / (4 * Precision + Recall)
print('F2:', F2)
F0_5 = 1.25 * (Precision * Recall) / (0.25 * Precision + Recall)
print('F0.5:', F0_5)
```

```
Accuracy:  0.8057212127151553
Error Rate: 0.19427878728848447
Sensitivity: 1.0
Specificity: 0.0
Precision: 0.8057212127151553
F1: 0.8924093121812977
F2: 0.9539939821455088
F0.5: 0.8382937571397135
```