In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
flights=pd.read_csv('flights.csv')
flights=flights.sample(n=100000)
flights.head()
```

```
C:\Users\SRAVIKA\AppData\Local\Temp\ipykernel_5204\664831390.py:1: DtypeWarning: Colu
mns (7,8) have mixed types. Specify dtype option on import or set low_memory=False.
  flights=pd.read_csv('flights.csv')
```

Out[2]:

	YEAR	MONTH	DAY	DAY_OF_WEEK	AIRLINE	FLIGHT_NUMBER	TAIL_NUMBER	ORIGIN_AI
3176246	2015	7	18	6	OO	4530	N437SW	
2242266	2015	5	22	5	MQ	3027	N699MQ	
1320012	2015	3	26	4	WN	588	N7740A	
4798016	2015	10	27	2	EV	4327	N21154	
1169211	2015	3	17	2	AA	2399	N566AA	

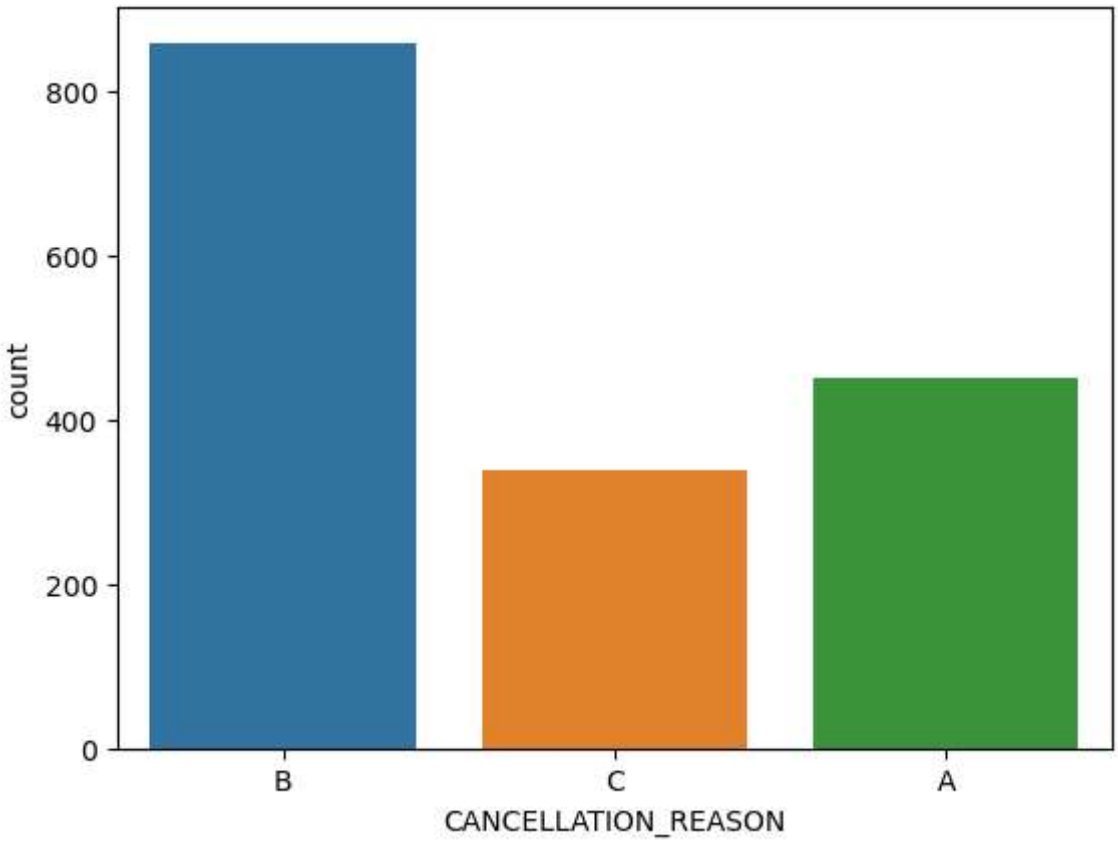
5 rows × 31 columns

In [3]:
```
flights.shape
```

Out[3]:
```
(100000, 31)
```

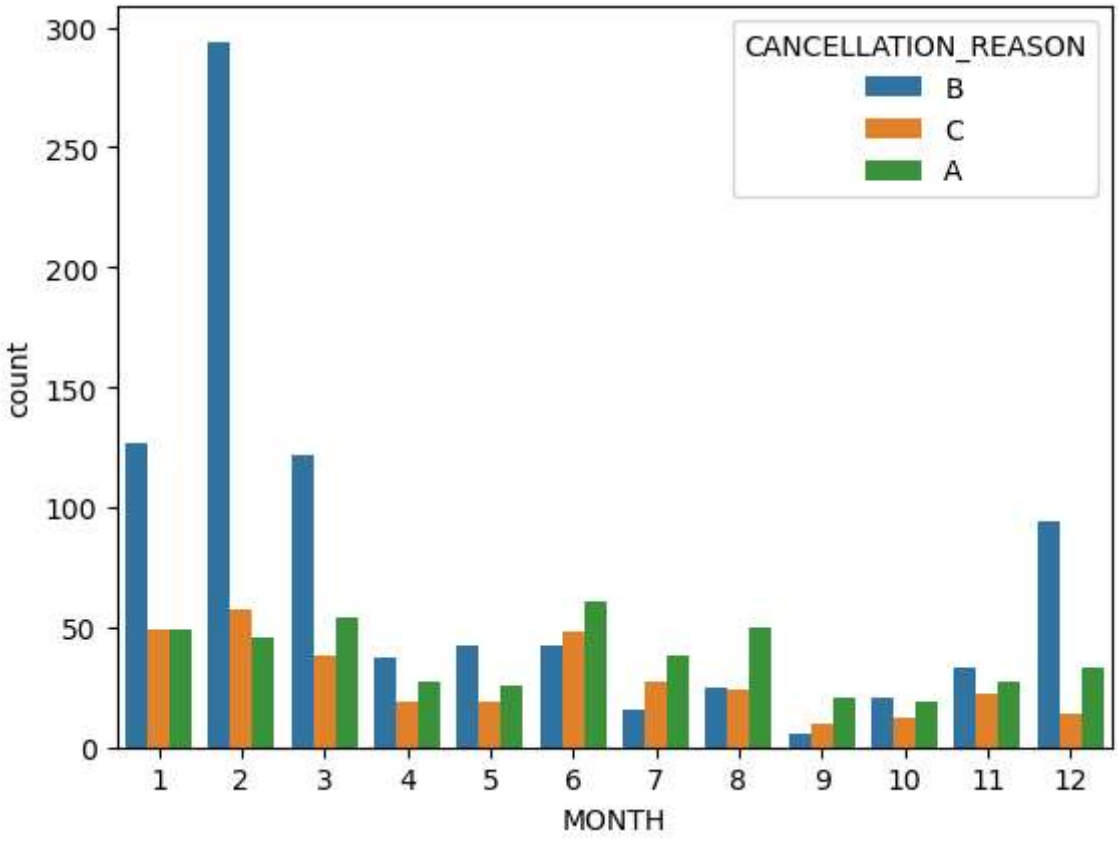
In [4]:
```
sns.countplot(x='CANCELLATION_REASON',data=flights)
```

Out[4]:
```
<AxesSubplot:xlabel='CANCELLATION_REASON', ylabel='count'>
```

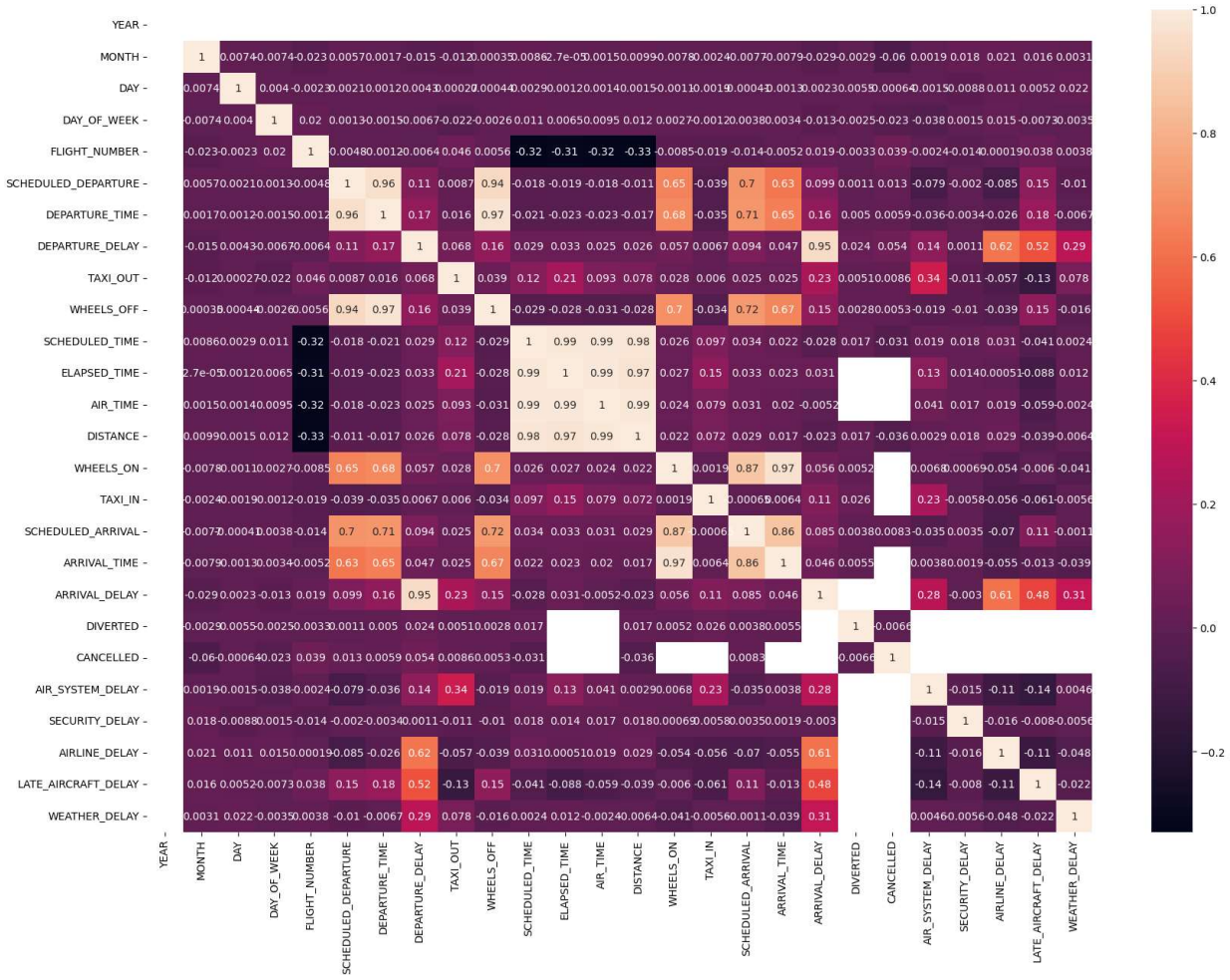


In [5]:
```
sns.countplot(x="MONTH",hue="CANCELLATION_REASON",data=flights)
```

Out[5]:
```
<AxesSubplot:xlabel='MONTH', ylabel='count'>
```



In [6]:
```
axis = plt.subplots(figsize=(20,14))
sns.heatmap(flights.corr(),annot = True)
plt.show()
```



In [7]:
```
corr=flights.corr()
corr
```

Out[7]:

	YEAR	MONTH	DAY	DAY_OF_WEEK	FLIGHT_NUMBER	SCHEDULED_D
YEAR	NaN	NaN	NaN	NaN	NaN	
MONTH	NaN	1.000000	0.007404	-0.007435	-0.023425	
DAY	NaN	0.007404	1.000000	0.003995	-0.002314	
DAY_OF_WEEK	NaN	-0.007435	0.003995	1.000000	0.019925	
FLIGHT_NUMBER	NaN	-0.023425	-0.002314	0.019925	1.000000	
SCHEDULED_DEPARTURE	NaN	0.005749	0.002132	0.001283	-0.004834	
DEPARTURE_TIME	NaN	0.001698	0.001201	-0.001541	-0.001158	
DEPARTURE_DELAY	NaN	-0.015025	0.004262	-0.006698	-0.006419	
TAXI_OUT	NaN	-0.011726	0.000275	-0.022092	0.046290	
WHEELS_OFF	NaN	0.000348	0.000445	-0.002570	0.005564	
SCHEDULED_TIME	NaN	0.008612	0.002934	0.010537	-0.315476	
ELAPSED_TIME	NaN	-0.000027	0.001181	0.006532	-0.305228	
AIR_TIME	NaN	0.001548	0.001429	0.009544	-0.317976	
DISTANCE	NaN	0.009926	0.001530	0.011717	-0.330073	
WHEELS_ON	NaN	-0.007752	-0.001115	0.002743	-0.008461	
TAXI_IN	NaN	-0.002436	-0.001866	-0.001212	-0.019084	
SCHEDULED_ARRIVAL	NaN	-0.007672	-0.000406	0.003775	-0.013788	
ARRIVAL_TIME	NaN	-0.007876	-0.001349	0.003438	-0.005175	
ARRIVAL_DELAY	NaN	-0.028645	0.002332	-0.012582	0.019035	
DIVERTED	NaN	-0.002915	0.005524	-0.002538	-0.003347	
CANCELLED	NaN	-0.059548	-0.000642	-0.023461	0.038922	
AIR_SYSTEM_DELAY	NaN	0.001889	-0.001466	-0.037736	-0.002376	
SECURITY_DELAY	NaN	0.018196	-0.008788	0.001511	-0.014337	
AIRLINE_DELAY	NaN	0.020629	0.010997	0.015137	0.000192	
LATE_AIRCRAFT_DELAY	NaN	0.015937	0.005163	-0.007295	0.038434	
WEATHER_DELAY	NaN	0.003119	0.021716	-0.003518	0.003780	

26 rows × 26 columns

In [8]:
```
variables_to_remove=["YEAR","FLIGHT_NUMBER","TAIL_NUMBER","DEPARTURE_TIME","TAXI_OUT",
flights.drop(variables_to_remove,axis=1,inplace= True)
flights.columns
```

Out[8]:
```
Index(['MONTH', 'DAY', 'DAY_OF_WEEK', 'AIRLINE', 'ORIGIN_AIRPORT',
       'DESTINATION_AIRPORT', 'SCHEDULED_DEPARTURE', 'DEPARTURE_DELAY',
       'DISTANCE', 'ARRIVAL_DELAY'],
      dtype='object')
```

In [9]:
```
airport = pd.read_csv('airports.csv')
airport
```

Out[9]:

	IATA_CODE	AIRPORT	CITY	STATE	COUNTRY	LATITUDE	LONGITUDE
0	ABE	Lehigh Valley International Airport	Allentown	PA	USA	40.65236	-75.44040
1	ABI	Abilene Regional Airport	Abilene	TX	USA	32.41132	-99.68190
2	ABQ	Albuquerque International Sunport	Albuquerque	NM	USA	35.04022	-106.60919
3	ABR	Aberdeen Regional Airport	Aberdeen	SD	USA	45.44906	-98.42183
4	ABY	Southwest Georgia Regional Airport	Albany	GA	USA	31.53552	-84.19447
...	...	...	...	...	...	...	...
317	WRG	Wrangell Airport	Wrangell	AK	USA	56.48433	-132.36982
318	WYS	Westerly State Airport	West Yellowstone	MT	USA	44.68840	-111.11764
319	XNA	Northwest Arkansas Regional Airport	Fayetteville/Springdale/Rogers	AR	USA	36.28187	-94.30681
320	YAK	Yakutat Airport	Yakutat	AK	USA	59.50336	-139.66023
321	YUM	Yuma International Airport	Yuma	AZ	USA	32.65658	-114.60597

322 rows × 7 columns

In [10]:
```
flights.loc[~flights.ORIGIN_AIRPORT.isin(airport.IATA_CODE.values),'ORIGIN_AIRPORT']='
flights.loc[~flights.DESTINATION_AIRPORT.isin(airport.IATA_CODE.values),'DESTINATION_A
flights
```

Out[10]:

	MONTH	DAY	DAY_OF_WEEK	AIRLINE	ORIGIN_AIRPORT	DESTINATION_AIRPORT	SCHEDU
3176246	7	18	6	OO	FSD	MSP	
2242266	5	22	5	MQ	BWI	ORD	
1320012	3	26	4	WN	MSY	BNA	
4798016	10	27	2	EV	OTHER	OTHER	
1169211	3	17	2	AA	DFW	IAH	
...	...	...	...	...	...	...	
2831932	6	27	6	AA	KOA	LAX	
1908120	5	2	6	WN	BWI	PBI	
96068	1	7	3	DL	TPA	JFK	
1486098	4	6	1	AA	MFE	DFW	
5507869	12	11	5	AA	PHL	SJU	

100000 rows × 10 columns

In [11]:
```
print(flights.ORIGIN_AIRPORT.nunique())
print(flights.DESTINATION_AIRPORT.nunique())
print(flights.AIRLINE.nunique())
```

```
323
322
14
```

In [12]:
```
flights=flights.dropna()
flights
```

Out[12]:

	MONTH	DAY	DAY_OF_WEEK	AIRLINE	ORIGIN_AIRPORT	DESTINATION_AIRPORT	SCHEDU
3176246	7	18	6	OO	FSD	MSP	
2242266	5	22	5	MQ	BWI	ORD	
1320012	3	26	4	WN	MSY	BNA	
4798016	10	27	2	EV	OTHER	OTHER	
1169211	3	17	2	AA	DFW	IAH	
...	...	...	...	...	...	...	
2831932	6	27	6	AA	KOA	LAX	
1908120	5	2	6	WN	BWI	PBI	
96068	1	7	3	DL	TPA	JFK	
1486098	4	6	1	AA	MFE	DFW	
5507869	12	11	5	AA	PHL	SJU	

98089 rows × 10 columns

In [13]:
```
flights.shape
```

Out[13]:
```
(98089, 10)
```

In [14]:
```
df=pd.DataFrame(flights)
df['DAY_OF_WEEK']= df['DAY_OF_WEEK'].apply(str)
df["DAY_OF_WEEK"].replace({"1":"SUNDAY", "2": "MONDAY", "3": "TUESDAY", "4":"WEDNESDAY
flights
```

Out[14]:

	MONTH	DAY	DAY_OF_WEEK	AIRLINE	ORIGIN_AIRPORT	DESTINATION_AIRPORT	SCHEDU
3176246	7	18	FRIDAY	OO	FSD	MSP	
2242266	5	22	THURSDAY	MQ	BWI	ORD	
1320012	3	26	WEDNESDAY	WN	MSY	BNA	
4798016	10	27	MONDAY	EV	OTHER	OTHER	
1169211	3	17	MONDAY	AA	DFW	IAH	
...	...	...	...	...	...	...	
2831932	6	27	FRIDAY	AA	KOA	LAX	
1908120	5	2	FRIDAY	WN	BWI	PBI	
96068	1	7	TUESDAY	DL	TPA	JFK	
1486098	4	6	SUNDAY	AA	MFE	DFW	
5507869	12	11	THURSDAY	AA	PHL	SJU	

98089 rows × 10 columns

In [15]:
```
dums = ['AIRLINE','ORIGIN_AIRPORT','DESTINATION_AIRPORT','DAY_OF_WEEK']
df_cat=pd.get_dummies(df[dums],drop_first=True)
df_cat
```

Out[15]:

	AIRLINE_AS	AIRLINE_B6	AIRLINE_DL	AIRLINE_EV	AIRLINE_F9	AIRLINE_HA	AIRLINE_MQ
3176246	0	0	0	0	0	0	0
2242266	0	0	0	0	0	0	1
1320012	0	0	0	0	0	0	0
4798016	0	0	0	1	0	0	0
1169211	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...
2831932	0	0	0	0	0	0	0
1908120	0	0	0	0	0	0	0
96068	0	0	1	0	0	0	0
1486098	0	0	0	0	0	0	0
5507869	0	0	0	0	0	0	0

98089 rows × 662 columns

In [16]:
```
df_cat.columns
```

Out[16]:
```
Index(['AIRLINE_AS', 'AIRLINE_B6', 'AIRLINE_DL', 'AIRLINE_EV', 'AIRLINE_F9',
       'AIRLINE_HA', 'AIRLINE_MQ', 'AIRLINE_NK', 'AIRLINE_OO', 'AIRLINE_UA',
       ...
       'DESTINATION_AIRPORT_WYS', 'DESTINATION_AIRPORT_XNA',
       'DESTINATION_AIRPORT_YAK', 'DESTINATION_AIRPORT_YUM',
       'DAY_OF_WEEK_MONDAY', 'DAY_OF_WEEK_SATURDAY', 'DAY_OF_WEEK_SUNDAY',
       'DAY_OF_WEEK_THURSDAY', 'DAY_OF_WEEK_TUESDAY', 'DAY_OF_WEEK_WEDNESDAY'],
      dtype='object', length=662)
```

In [17]:
```
df.columns
```

Out[17]:
```
Index(['MONTH', 'DAY', 'DAY_OF_WEEK', 'AIRLINE', 'ORIGIN_AIRPORT',
       'DESTINATION_AIRPORT', 'SCHEDULED_DEPARTURE', 'DEPARTURE_DELAY',
       'DISTANCE', 'ARRIVAL_DELAY'],
      dtype='object')
```

In [18]:
```
flights.columns
```

Out[18]:
```
Index(['MONTH', 'DAY', 'DAY_OF_WEEK', 'AIRLINE', 'ORIGIN_AIRPORT',
       'DESTINATION_AIRPORT', 'SCHEDULED_DEPARTURE', 'DEPARTURE_DELAY',
       'DISTANCE', 'ARRIVAL_DELAY'],
      dtype='object')
```

In [19]:
```
var_to_remove=["DAY_OF_WEEK","AIRLINE","ORIGIN_AIRPORT","DESTINATION_AIRPORT"]
df.drop(var_to_remove,axis=1,inplace=True)
df
```

Out[19]:

	MONTH	DAY	SCHEDULED_DEPARTURE	DEPARTURE_DELAY	DISTANCE	ARRIVAL_DELAY
3176246	7	18	815	-9.0	196	-6.0
2242266	5	22	1754	-8.0	622	5.0
1320012	3	26	1915	21.0	471	15.0
4798016	10	27	700	19.0	645	7.0
1169211	3	17	1625	-3.0	224	26.0
...	...	...	...	...	...	...
2831932	6	27	1310	-16.0	2504	11.0
1908120	5	2	815	8.0	883	-1.0
96068	1	7	705	-6.0	1005	-5.0
1486098	4	6	759	-1.0	469	0.0
5507869	12	11	1805	-8.0	1576	12.0

98089 rows × 6 columns

In [20]:
```
data=pd.concat([df,df_cat],axis=1)
data
```

Out[20]:

	MONTH	DAY	SCHEDULED_DEPARTURE	DEPARTURE_DELAY	DISTANCE	ARRIVAL_DELAY	AI
3176246	7	18	815	-9.0	196	-6.0	
2242266	5	22	1754	-8.0	622	5.0	
1320012	3	26	1915	21.0	471	15.0	
4798016	10	27	700	19.0	645	7.0	
1169211	3	17	1625	-3.0	224	26.0	
...	...	...	...	...	...	...	
2831932	6	27	1310	-16.0	2504	11.0	
1908120	5	2	815	8.0	883	-1.0	
96068	1	7	705	-6.0	1005	-5.0	
1486098	4	6	759	-1.0	469	0.0	
5507869	12	11	1805	-8.0	1576	12.0	

98089 rows × 668 columns

In [21]:
```
final_data = data.sample(n=60000)
final_data
```

Out[21]:

	MONTH	DAY	SCHEDULED_DEPARTURE	DEPARTURE_DELAY	DISTANCE	ARRIVAL_DELAY	AI
2834199	6	27	1542	-2.0	177	-4.0	
307625	1	21	827	12.0	872	5.0	
1351373	3	28	1830	-3.0	1756	-32.0	
3090903	7	13	915	-2.0	369	-14.0	
827776	2	24	1147	0.0	1127	-21.0	
...	...	...	...	...	...	...	
1492857	4	6	1431	63.0	409	54.0	
345518	1	23	1440	51.0	1751	54.0	
1387458	3	31	600	-2.0	453	-13.0	
1999712	5	7	1949	4.0	930	-3.0	
4445343	10	4	1940	-3.0	748	-17.0	

60000 rows × 668 columns

In [22]:
```
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

In [23]:
```
X=final_data.drop("DEPARTURE_DELAY",axis=1)
Y=final_data.DEPARTURE_DELAY
```

In [24]:
```
X
```

Out[24]:

	MONTH	DAY	SCHEDULED_DEPARTURE	DISTANCE	ARRIVAL_DELAY	AIRLINE_AS	AIRLINE_I
2834199	6	27	1542	177	-4.0	0	
307625	1	21	827	872	5.0	0	
1351373	3	28	1830	1756	-32.0	0	
3090903	7	13	915	369	-14.0	0	
827776	2	24	1147	1127	-21.0	0	
...	...	...	...	...	...	...	
1492857	4	6	1431	409	54.0	0	
345518	1	23	1440	1751	54.0	0	
1387458	3	31	600	453	-13.0	0	
1999712	5	7	1949	930	-3.0	0	
4445343	10	4	1940	748	-17.0	0	

60000 rows × 667 columns

In [25]:
```
Y
```

Out[25]:
```
2834199    -2.0
307625     12.0
1351373    -3.0
3090903    -2.0
827776      0.0
           ...
1492857    63.0
345518     51.0
1387458    -2.0
1999712     4.0
4445343    -3.0
Name: DEPARTURE_DELAY, Length: 60000, dtype: float64
```

In [26]:
```
from sklearn.model_selection import train_test_split, cross_val_score, cross_val_predi
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=
```

In [27]:
```
from sklearn.ensemble import RandomForestRegressor
reg_rf = RandomForestRegressor()
reg_rf.fit(X_train,y_train)
```

Out[27]:
```
RandomForestRegressor()
```

In [28]:
```
y_pred = reg_rf.predict(X_test)
```

In [29]:
```
reg_rf.score(X_train,y_train)
```

Out[29]:
```
0.9900641038818335
```

In [30]:
```
reg_rf.score(X_test,y_test)
```

Out[30]:
```
0.9280426585481997
```

In [31]:
```
metrics.r2_score(y_test,y_pred)
```

Out[31]:
```
0.9280426585481997
```

In [32]:
```
print('MAE:', metrics.mean_absolute_error(y_test,y_pred))
print('MSE:', metrics.mean_squared_error(y_test,y_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
MAE: 6.018026666666666
MSE: 101.33330478333335
RMSE: 10.066444495616778
```

In [33]:
```
pp=pd.DataFrame({'Actual':y_test,'Predicted':y_pred})
pp
```

Out[33]:

	Actual	Predicted
720592	-4.0	-4.95
2435493	8.0	17.62
5726931	34.0	41.10
3591874	-9.0	-10.91
3574844	-1.0	-0.03
...	...	...
5552689	87.0	98.14
2315924	-8.0	-6.87
4848341	0.0	-1.17
522451	-2.0	-4.93
3699831	-6.0	-2.75

12000 rows × 2 columns

In [34]:
```
from sklearn.model_selection import RandomizedSearchCV
#Randomized Search CV

# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 10, stop = 200, num = 12)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10, 15, 100]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 5, 10]
```

In [35]:
```
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf}
```

In [36]:
```
rf_random = RandomizedSearchCV(estimator = reg_rf, param_distributions = random_grid,s
```

gradient boosting

In [42]:
```
from sklearn.ensemble import GradientBoostingRegressor
gbr=GradientBoostingRegressor(random_state=0)
```

In [43]:
```
GBR=gbr.fit(X_train,y_train)
pre=GBR.predict(X_test)
```

In [44]:
```
print('MAE:', metrics.mean_absolute_error(y_test,pre))
print('MSE:', metrics.mean_squared_error(y_test,pre))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test,pre)))
```

```
MAE: 6.0140912233718895
MSE: 96.20755799687015
RMSE: 9.808545151900466
```

In [45]:
```
metrics.r2_score(y_test,pre)
```

Out[45]:
```
0.9316824797550342
```

In [46]:
```
gg=pd.DataFrame({'Actual':y_test,'Predicted':pre})
gg
```

Out[46]:

	Actual	Predicted
720592	-4.0	-3.505726
2435493	8.0	16.150385
5726931	34.0	37.734716
3591874	-9.0	-8.610290
3574844	-1.0	1.290366
...	...	...
5552689	87.0	99.239412
2315924	-8.0	-5.921293
4848341	0.0	-3.678508
522451	-2.0	-5.691144
3699831	-6.0	-3.244165

12000 rows × 2 columns

In [47]:
```
def predict(MONTH, DAY,SCHEDULED_DEPARTURE,DISTANCE, ARRIVAL_DELAY,AIRLINE,ORIGIN_AIRP
    AIRLINE_index = np.where(X.columns==AIRLINE)[0][0]
    ORIGIN_index = np.where(X.columns==ORIGIN_AIRPORT)[0][0]
    DESTINATION_index = np.where(X.columns==DESTINATION_AIRPORT)[0][0]
    DAY_OF_WEEK_index = np.where(X.columns==DAY_OF_WEEK)[0][0]
    x= np.zeros(len(X.columns))
    x[0] = MONTH
    x[1] = DAY
    x[2] = SCHEDULED_DEPARTURE
    x[3] = DISTANCE
    x[4] = ARRIVAL_DELAY
    if AIRLINE_index >=0:
        x[AIRLINE_index] = 1
    if ORIGIN_index >=0:
        x[ORIGIN_index] = 1
    if DESTINATION_index >=0:
        x[DESTINATION_index] = 1
    if  DAY_OF_WEEK_index >= 0:
        x[ DAY_OF_WEEK_index] = 1

    return gbr.predict([x])[0]
```

In [48]:
```
res= predict(5,6,1515,328,-8.0,'AIRLINE_OO','ORIGIN_AIRPORT_PHX','DESTINATION_AIRPORT_
res
```

```
C:\Users\SRAVIKA\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does
not have valid feature names, but GradientBoostingRegressor was fitted with feature n
ames
  warnings.warn(
```

Out[48]:
```
-3.8612915448626035
```

In [49]:
```
if(res<=-15):
  print("Flight is delayed")
else:
  print("Flight is not delayed")
```

```
Flight is not delayed
```

In [ ]:
```

```