project-3-flight-status-prediction

March 31, 2024

1 Flights Delay Data Analysis

1.1 Table of contents

1. Case Study Summary
2. Data Source Description
3. Data Processing
4. Analyses and Visualisation
5. Observations and Conclusions
6. Flight Delay Predictions

1.2 1. Case Study Summary

Business Task:

Analyze flights data over 5 years (2018-2022) in order to gain insights about flight delay patterns.

- Predict which flights will be cancelled or delayed
- Predict the delay time?
- Explore how different airlines compare?

1.3 2. Data Source Description

This dataset contains all flight information including cancellation and delays by airline for dates back to January 2018.

Combined_Flights_XXXX.csv or Combined_Flights_XXXX.parquet files can be used in order to access the combined data for the entire year. These files also have filtered out columns that are mostly null in the original dataset.The raw data including all columns by month can be found in the files named Flights_XXXX_X.csv

The data is publicly stored at THIS source (CC0: Public Domain)

Import necessary libraries and the data files:

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

```
import warnings
import squarify
warnings.filterwarnings("ignore")
%matplotlib inline


parquet_files = glob("../input/flight-delay-dataset-20182022/*.parquet")
column_subset = [
    "FlightDate",
    "Airline",
    "Flight_Number_Marketing_Airline",
    "Origin",
    "Dest",
    "Cancelled",
    "Diverted",
    "CRSDepTime",
    "DepTime",
    "DepDelayMinutes",
    "OriginAirportID",
    "OriginCityName",
    "OriginStateName",
    "DestAirportID",
    "DestCityName",
    "DestStateName",
    "TaxiOut",
    "TaxiIn",
    "CRSArrTime",
    "ArrTime",
    "ArrDelayMinutes",
    "Year",
    "Month",
]

dfs = []
for f in parquet_files:
    dfs.append(pd.read_parquet(f, columns=column_subset))
flights = pd.concat(dfs).reset_index(drop=True)

cat_cols = ["Airline", "Origin", "Dest", "OriginStateName", "DestStateName"]
for c in cat_cols:
    flights[c] = flights[c].astype("category")
```

1.4 3.Data Processing

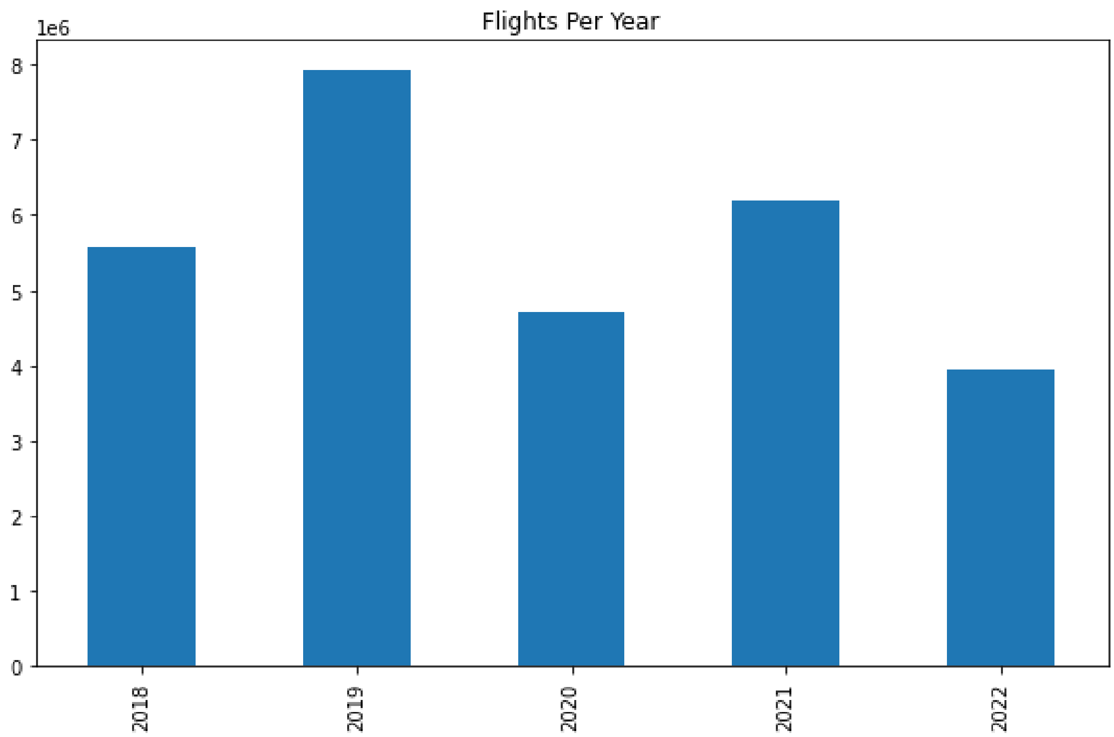
Data Cleaning:

[2]:
```
# Dropping duplicate rows
flights.drop_duplicates()

# Remove columns with null departure or arrival time
flights = flights[flights['DepTime'].notnull()]
flights = flights[flights['ArrTime'].notnull()]


# Remove null values from departure delay column
flights['DepTime'] = flights['DepTime'].astype(int).astype(str).str.zfill(4)
flights['CRSDepTime'] = flights['CRSDepTime'].astype(int).astype(str).str.
↪zfill(4)

flights = flights[flights['DepTime']!='2400']
flights['DepTimeDateTime'] = pd.to_datetime(flights['DepTime'], format="%H%M")
flights = flights[flights['CRSDepTime']!='2400']
flights['CRSDepTimeDateTime']= pd.to_datetime(flights['CRSDepTime'],␣
↪format="%H%M")

flights['DepDelay'] = (flights['DepTimeDateTime'] -␣
↪flights['CRSDepTimeDateTime']).astype('timedelta64[s]')/60
flights['DepDelayMinutes'] = flights['DepDelay'].apply(lambda x: x if x > 0␣
↪else 0)
flights.drop(columns =['DepTimeDateTime','CRSDepTimeDateTime'])


# Remove null values from arrival delay column
flights['ArrTime'] = flights['ArrTime'].astype(int).astype(str).str.zfill(4)
flights['CRSArrTime'] = flights['CRSArrTime'].astype(int).astype(str).str.
↪zfill(4)

flights = flights[flights['ArrTime']!='2400']
flights['ArrTimeDateTime'] = pd.to_datetime(flights['ArrTime'], format="%H%M")
flights = flights[flights['CRSArrTime']!='2400']
flights['CRSArrTimeDateTime']= pd.to_datetime(flights['CRSArrTime'],␣
↪format="%H%M")

flights['ArrDelay'] = (flights['ArrTimeDateTime'] -␣
↪flights['CRSArrTimeDateTime']).astype('timedelta64[s]')/60
flights['ArrDelayMinutes'] = flights['ArrDelay'].apply(lambda x: x if x > 0␣
↪else 0)
flights.drop(columns =['ArrTimeDateTime','CRSArrTimeDateTime'])


# Remove null values from Taxi In/Taxi out columns
flights = flights[flights['TaxiIn'].notnull()]
```

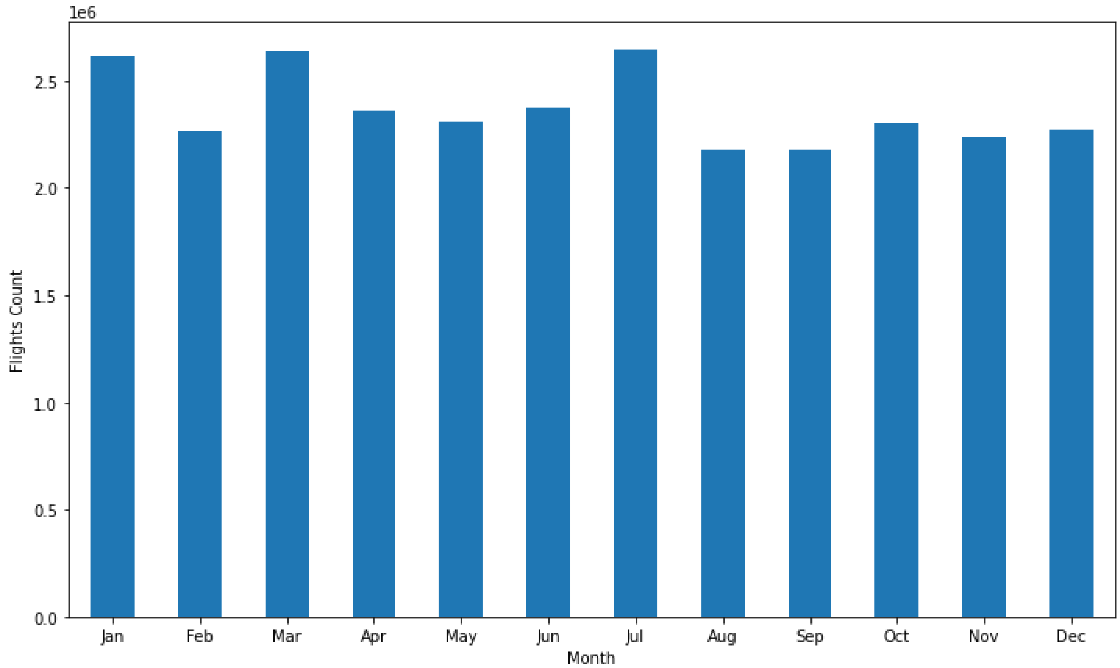
```
flights = flights[flights['TaxiOut'].notnull()]
```

1.5 4. Analysis and Visualisations

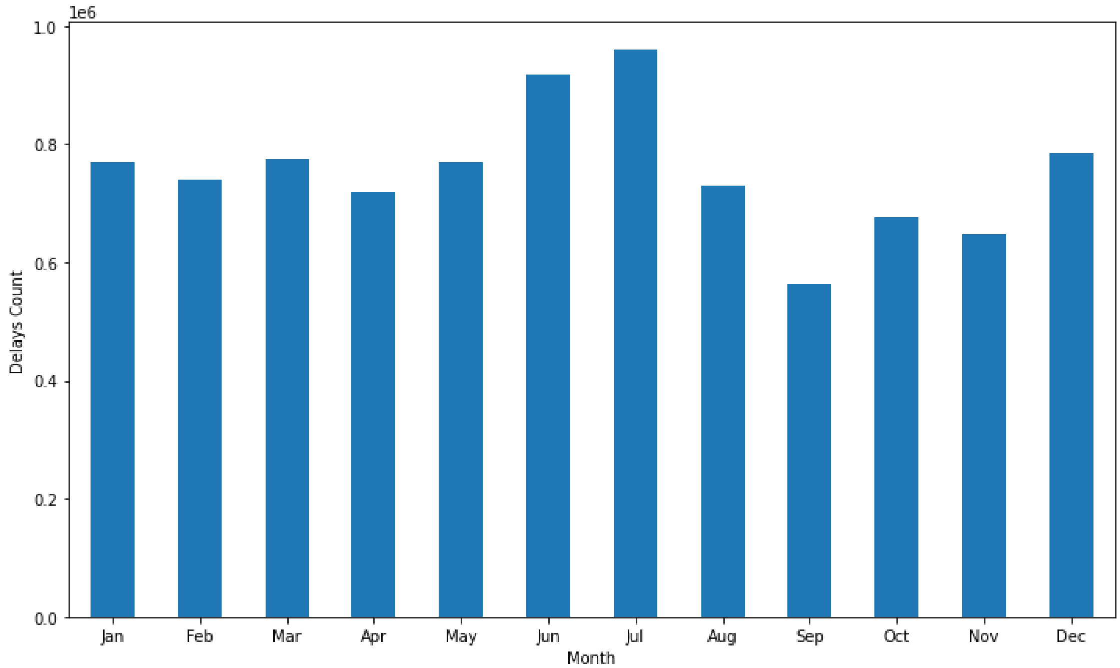
[3]:
```
# Flight count per year
flights["Year"].value_counts().sort_index().plot(
    kind="bar", figsize=(10, 6), title="Flights Per Year"
)
plt.show()
```



[4]:
```
# Total flights per month
flights_per_month = flights.groupby('Month',as_index=True).agg({'FlightDate':
 ↪'count'})
flights_per_month.index =␣
 ↪['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']
flights_per_month.plot(kind =␣
 ↪'bar',figsize=(12,7),rot=0,legend=False,ylabel='Flights␣
 ↪Count',xlabel='Month')
plt.show()
```

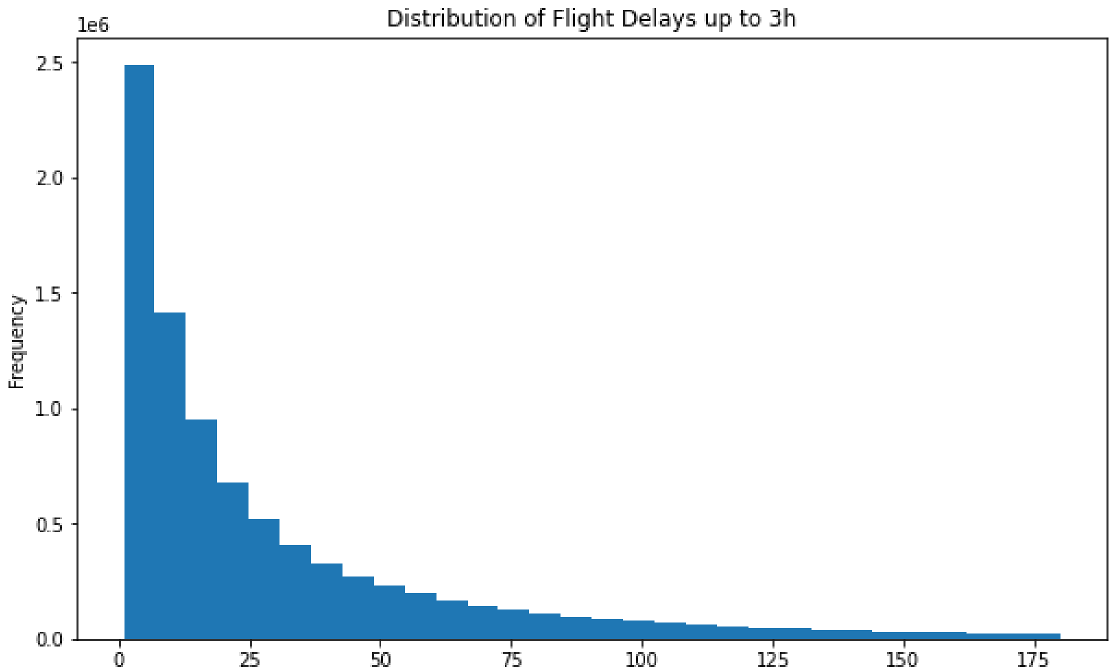


[5]:
```python
# Month with most delay
delays_per_month = flights[flights['DepDelay']>0].
↪groupby('Month',as_index=True).agg({'FlightDate':'count'})
delays_per_month.index =␣
↪['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']
delays_per_month.plot(kind =␣
↪'bar',figsize=(12,7),rot=0,legend=False,ylabel='Delays Count',xlabel='Month')
plt.show()
```



[6]:

```
# Delay time distribution
flights.query("DepDelayMinutes > 0 and DepDelayMinutes <␣
↪181")["DepDelayMinutes"].plot(
    kind="hist", bins=30, title="Distribution of Flight Delays up to 3h",␣
↪figsize=(10,6)
)
plt.show()
```



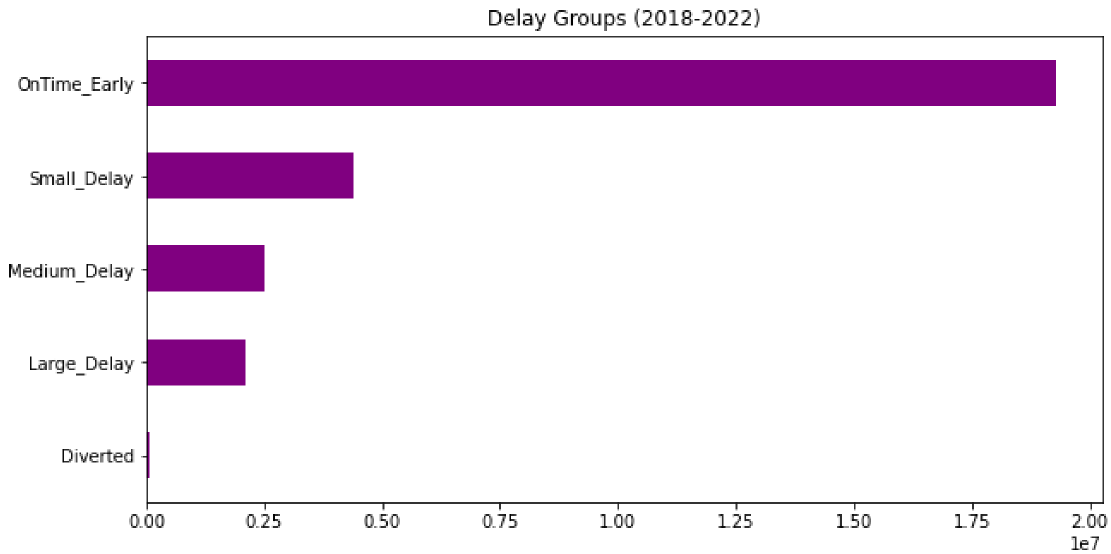
[7]:
```
delays0to25 = flights.query("DepDelayMinutes > 0 and DepDelayMinutes <=␣
↪25")["DepDelayMinutes"].count()
delays_total = flights.query("DepDelayMinutes > 0")["DepDelayMinutes"].count()

delays0to25/delays_total*100
```

[7]: 62.19845958720332

[8]:
```
flights["DelayGroup"] = None
flights.loc[flights["DepDelayMinutes"] == 0, "DelayGroup"] = "OnTime_Early"
flights.loc[
    (flights["DepDelayMinutes"] > 0) & (flights["DepDelayMinutes"] <= 15),␣
↪"DelayGroup"
] = "Small_Delay"
flights.loc[
    (flights["DepDelayMinutes"] > 15) & (flights["DepDelayMinutes"] <= 45),␣
↪"DelayGroup"
] = "Medium_Delay"
flights.loc[flights["DepDelayMinutes"] > 45, "DelayGroup"] = "Large_Delay"
flights.loc[flights["Cancelled"], "DelayGroup"] = "Cancelled"
flights.loc[flights["Diverted"], "DelayGroup"] = "Diverted"
```

[9]:
```
# Delay time grouping
flights["DelayGroup"].value_counts(ascending=True).plot(
```

```
    kind="barh", figsize=(10, 5), color='purple', title="Delay Groups␣
↪(2018-2022)"
)
plt.show()
```



[10]:
```
# % of delayed flights
flights_on_time = flights[flights["DelayGroup"]=='OnTime_Early']['FlightDate'].
↪count()
total_flights = flights['FlightDate'].count()

100 - flights_on_time/total_flights*100
```

[10]: 31.99460403389051

[11]:
```
# Delay time grouping per year
flights_agg = flights.groupby("Year")["DelayGroup"].
↪value_counts(normalize=True).unstack() * 100
col_order = ["OnTime_Early", "Small_Delay", "Medium_Delay", "Large_Delay",␣
↪"Diverted"] #Add cancelled as well
flights_agg[col_order].style.background_gradient(cmap="Purples")
```

[11]: <pandas.io.formats.style.Styler at 0x7f92df0f8c10>

[12]:
```
# Delay time grouping per month
flights_agg = flights.groupby("Month")["DelayGroup"].
↪value_counts(normalize=True).unstack() * 100
col_order = ["OnTime_Early", "Small_Delay", "Medium_Delay", "Large_Delay",␣
↪"Diverted"] #Add cancelled as well
```

```
flights_agg[col_order].style.background_gradient(cmap="Purples")
```

[12]: <pandas.io.formats.style.Styler at 0x7f9316ec8fd0>

[13]:
```
flights_per_airline = flights.groupby('Airline',as_index=True).
 ↪agg({'FlightDate':'count'})
top_airlines = flights_per_airline[flights_per_airline['FlightDate']>500000].
 ↪index
top_airlines.tolist()

flights_top_airlines = flights.loc[flights["Airline"].isin(top_airlines)].
 ↪reset_index(drop=True)
flights_top_airlines["Airline"] = flights_top_airlines["Airline"].astype("str").
 ↪astype("category")
```

[ ]:
```
# Top Airlines Flight Delay Time Breakdown

col_order = ["OnTime_Early", "Small_Delay", "Medium_Delay", "Large_Delay",␣
 ↪"Diverted"]

flights_agg = (
    flights_top_airlines.groupby(["Airline"])["DelayGroup"]
    .value_counts(normalize=True)
    .unstack()[col_order]
)

fig, ax = plt.subplots(figsize=(10, 5))
flights_agg.sort_values("OnTime_Early").plot(
    kind="barh", stacked=True, ax=ax, width=0.8, edgecolor="black"
)
ax.legend(bbox_to_anchor=(1, 1))
ax.set_title("Top Airlines Flight Delay Time Breakdown", fontsize=20)
ax.set_xlabel("Percent of Total Flights")
plt.show()
```

[ ]:
```
# Daily count of delays

# !pip install calmap plotly_calplot -q
import calmap

events = flights[flights["DepDelayMinutes"] > 0].
 ↪groupby("FlightDate")["DepDelayMinutes"].count()

fig, axs = plt.subplots(5, 1, figsize=(14, 14))
for i, year in enumerate([2018, 2019, 2020, 2021, 2022]):
    calmap.yearplot(
```

```
        events.apply(np.log), year=year, cmap="YlGn", monthly_border=True,␣
 ↪ax=axs[i]
    )
    axs[i].set_title(year)
fig.patch.set_facecolor("white")
fig.suptitle("Daily Delay Count", y=0.92, fontsize=20)
```

1.6 5. Observations and Conclusions

- Excluding 2022 (which is still ongoing), 2020 had the least flights. This is most likely due to global pandemic that started during that year and influenced the traveling
- It is more likely to have a delay during summer months(June, July). July also has the most total flights so there probably is a correlation.
- It is less likely to have a delay during the month of September
- Most delays are short ones, big majority falls into 0-25 minutes (62%)
- 31% of the flights are delayed or diverted
- Southwest Airlines company faced most delays out of the top companies while Republic Airlines the least.
- JetBlue Airways has the most large delays count
- Starting with March 2020 the amount of daily delays considerably dropped and kept a low trend until Apr-May 2021. This period coincides with the lock down and travel restrictions period. Fewer flights probably were correlated to higher capability of handling flights on time

[ ]:
```
# Predictions about flight delays
flights['delay'] = flights['DepDelay'].apply(lambda x: 1 if x > 0 else 0)
flights['big_delay'] = flights['DepDelay'].apply(lambda x: 1 if x >= 60 else 0)


from sklearn.ensemble import RandomForestRegressor,RandomForestClassifier
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split

from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from xgboost import XGBRegressor,XGBClassifier
from sklearn.preprocessing import LabelEncoder


labelencoder = LabelEncoder()
flights['Airline_Cat'] = labelencoder.fit_transform(flights['Airline'])
flights['OriginStateName_Cat'] = labelencoder.
 ↪fit_transform(flights['OriginStateName'])
flights['DestStateName_Cat'] = labelencoder.
 ↪fit_transform(flights['DestStateName'])
flights['Origin_Cat'] = labelencoder.fit_transform(flights['Origin'])
```

```
flights['Dest_Cat'] = labelencoder.fit_transform(flights['Dest'])

smaller_sample = flights.sample(n=15000000)

y = smaller_sample["delay"]
features =␣
 ↪['Month','Airline_Cat','OriginStateName_Cat','DestStateName_Cat','Origin_Cat','Dest_Cat']
X = smaller_sample[features]

train_X, val_X, train_y, val_y = train_test_split(X, y, random_state = 1)

model = XGBClassifier(n_estimators=1000, learning_rate=0.05)

model.fit(train_X, train_y,
             early_stopping_rounds=5,
             eval_set=[(val_X, val_y)],
             verbose=False)

val_predictions = model.predict(val_X)
mean_absolute_error(val_predictions, val_y)

# 0.309
# The first model returned 0.336. That model used the one hot encoder and only␣
 ↪had the 'Month' and 'Airline' features.
```

1. what is the chance of having no delay?
2. chance of having a delay less than 30 min?
3. chance of having a delay if u fly with X company?
4. chance of having a delay if u fly in june? how about september?

[ ]:
```
# What is the chance of having no delay?
delays_total = flights.query("DepDelayMinutes > 0")["DepDelayMinutes"].count()
total_flights = flights['FlightDate'].count()

delay_chance = delays_total/total_flights*100

# 31.881539858398167
```

[ ]:
```
# Chance of having a delay less than 30 min?
delays0to30 = flights.query("DepDelayMinutes > 0 and DepDelayMinutes <=␣
 ↪30")["DepDelayMinutes"].count()
delays_total = flights.query("DepDelayMinutes > 0")["DepDelayMinutes"].count()

less_than_30_min_delay_chance = delays0to30/delays_total*100

# P(delay)*P(delay[0:30])
```

```
delay_chance/100*less_than_30_min_delay_chance/100
# 21.31%
```

[ ]:
```
# Chance of having a delay if u fly with X company? How about having a large␣
↪delay? (> 45 min)
company_flights = flights[flights['Airline'] == 'JetBlue Airways']
total_delays_from_company = company_flights.query("DepDelayMinutes >␣
↪0")["DepDelayMinutes"].count()
total_flights_of_company = flights[flights['Airline'] == 'JetBlue␣
↪Airways']['FlightDate'].count()

total_delays_from_company/total_flights_of_company*100
# 38.33%

# Large Delay
large_delays_from_company = company_flights.query("DepDelayMinutes >␣
↪45")["DepDelayMinutes"].count()
large_delays_from_company/total_flights_of_company*100
# 12.21%
```

[ ]:
```
# Chance of having a delay if u fly in june?
june_flights = flights[flights['Month'] == 6]
total_delays_june = june_flights.query("DepDelayMinutes >␣
↪0")["DepDelayMinutes"].count()
total_flights_june = flights[flights['Month'] == 6]['FlightDate'].count()


total_delays_june / total_flights_june *100
# 38.67%
```