Seminar A: Introduction to practical IoT design and applied machine learning

Environment and Information Studies 2nd year
Doyoon Lee (student number)
Yoshiyasu Takefuji

December 20, 2020

Which services in what order do airline companies in the United States have to improve in order to attract as many passengers having not traveled from the beginning of the year due to Covid-19 as possible from now.

Covid-19 is the pandemic infectious disease caused by the most recently discovered coronavirus, significantly affecting all the global companies and people in 2020. From the various types of industries, transportation industry is the one of the most damaged industries in the world. Especially, airline industry which gets much of its profit from passengers is the worst since most of the countries have restricted entrance of tourists while people inevitably or sometimes use their domestic transportations. Even before Covid-19, some of the largest airlines had suffered from financial balances therefore the outbreak has worsened their situations.

Under the current situation, airline companies in the US have to improve their insufficient services now in order to attract passengers, who have not used airlines during this covid-19 situation, after covid-19 settled down. In the US, there are a number of airlines competing to try to attract more passengers by providing various services and marketing. After this pandemic situation, it is obvious that passengers having not traveled overseas until the time when Covid-19 settled down would use airlines more for various purposes, traveling or meeting families living in foreign countries for example. Airlines should prepare the situation by improving their services to attract more passengers to their airlines. It is directly related with their profits which have been extremely decreased in 2020. It would be better to improve all the services affecting satisfaction of passengers, but time is limited. They have to think of the order of priority and improve from the first place, which can be realized through machine learning model from a survey data whether a passenger satisfied or not based on various factors such as age, WIFI service in an aviation, online booking service, and so on. This research would be the one of solutions for all the airlines not only in the US but also in the world how to face and prepare in the near future situation.

This research firstly deals with trends of Covid-19 new cases and airport traffic in the US in order to examine traffic trends in the near future and predict the time when passengers start to use airlines as like before the outbreak of Covid-19. And then, the order from the most to worst importance for affecting passenger satisfaction on airline experience is found through data of passenger satisfaction survey.

-United States Daily Number of New Cases / Trend-

Data is from Statista website (https://www.statista.com/statistics/1102816/coronavirus-covid19-cases-number-us-americans-by-day/), originally surveyed by and published by WHO. The data is the number of daily new cases of Covid-19 in the US from January 20, 2020 to December 13, 2020.

Importing the data file downloaded from the website by pandas library and taking a glance at it.

In [1]:
```
# data from: https://www.statista.com/statistics/1102816/coronavirus-covid19-cases-number-us-americans-by-day/
import pandas as pd

df=pd.read_excel("us_cases.xlsx",sheet_name='Data')
df.head()
```

Out[1]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2
0	NaN	NaN	NaN
1	NaN	Number of U.S. coronavirus (COVID-19) cases fr...	NaN
2	NaN	Number of new cases of coronavirus (COVID-19) ...	NaN
3	NaN	NaN	NaN
4	NaN	Jan 20	5.0

Saving a necessary data into variable x and y and checking the data to make sure proper data is saved.

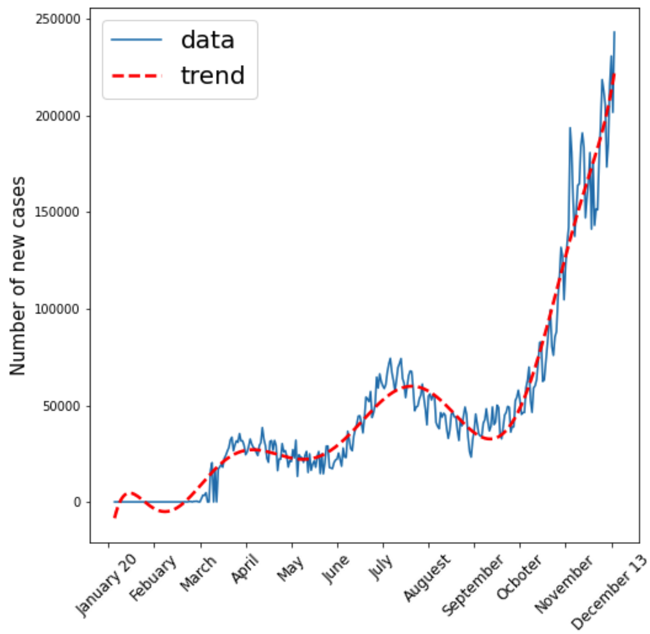
In [2]:
```
import numpy as np

n=len(df["Unnamed: 2"])
y=df["Unnamed: 2"][4:n]
y=y.replace("-",0)
x=np.arange(4,n)
y.head()
```

Out[2]:
```
4    5.0
5    0.0
6    0.0
7    0.0
8    1.0
Name: Unnamed: 2, dtype: float64
```

Plotting number of new cases (y) versus time (x) as a line and a trend line made by polyfit( ) function. The degree of the fitting polynomial is set as 10.

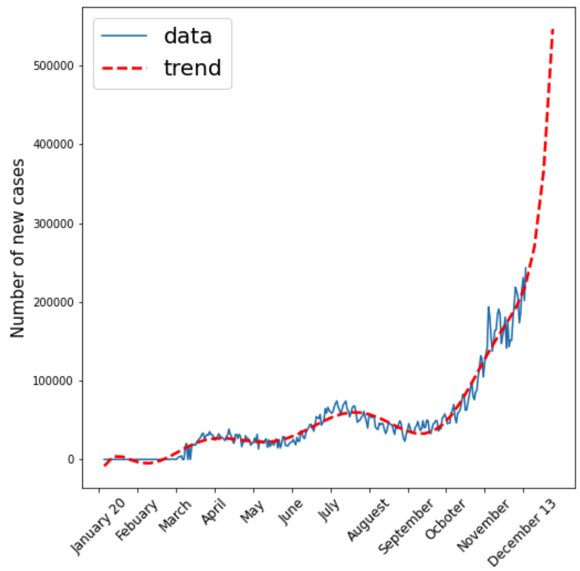
```
In [3]: trendpoly=np.poly1d(np.polyfit(x,y,10))
```

```
In [4]: import matplotlib.pyplot as plt

        plt.figure(figsize=(8,8))
        plt.plot(x,y,label='data')
        plt.plot(x,trendpoly(x),'--r',label='trend',linewidth=2.5)
        plt.xticks(np.arange(0,360,30),labels=["January 20","Febuary","March","April",
                                               "May","June","July","Auguest","September",
                                               "Ocboter","November","December 13"],
                                               rotation=45,fontsize=12)
        plt.ylabel("Number of new cases",fontsize=15)
        plt.legend(fontsize=20)
        plt.show()
```



Extending the trend line to see the trend in the future. The trend line in the graph below shows the number of new cases in the US will continue to increase.

```
In [5]: x_new=np.linspace(4,n+20)

        plt.figure(figsize=(8,8))
        plt.plot(x,y,label='data')
        plt.plot(x_new,trendpoly(x_new),'--r',label='trend',linewidth=2.5)
        plt.xticks(np.arange(0,360,30),labels=["January 20","Febuary","March","April",
                                               "May","June","July","Auguest","September",
                                               "Ocboter","November","December 13"],
                                               rotation=45,fontsize=12)
        plt.ylabel("Number of new cases",fontsize=15)
        plt.legend(fontsize=20)
        plt.show()
```



-United States Airport Traffic / Trend-

Data is from TSA (Transportation Security Administration) website, an official website of the US government (https://www.tsa.gov/coronavirus/passenger-throughput?page=0). The data is daily number of passengers screened at TSA checkpoints in the US from March 1, 2020 to December 13, 2020. Instead of searching for pre-existed data file, the data is collected by web crawling and scraping technique in order to get the latest data updated on the website.

Extracting and saving the data into csv file are done by csv, requests, beautifulsoup4, and re libraries. Because the latest data is appended to the top of the data list on the website, the data is initially extracted from page 1 and then page 0. Regular expression helps extract the only necessary data element on the website. The extracted date and number data are saved into csv file to be used for plotting a graph.

```python
# data from: https://www.tsa.gov/coronavirus/passenger-throughput?page=0

import csv
import requests
from bs4 import BeautifulSoup as bs
import re

for i in range(1,-1,-1):
    page = requests.get(f'https://www.tsa.gov/coronavirus/passenger-throughput?page={i}')
    soup = bs(page.text, 'html.parser')

    d = re.compile("\d+/\d+")
    date = soup.findAll("td",{"class": "views-field views-field-field-today-date"})
    date_data = [d.match(element.text.strip()).group() for element in date]

    number = soup.findAll("td",{"views-field views-field-field-this-year"})
    num_data = [element.text.strip().replace(",","") for element in number]

    f = open('tsa_traffic_data.csv','w',newline='')
    if i==1:
        wr = csv.writer(f)
        wr.writerow(["Date","Number of passengers"])

    for j in range(len(date_data)-1,-1,-1):
        data=[date_data[j],num_data[j]]
        wr.writerow(data)
f.close()
```

Importing the scraped data by pandas library and taking a glance at it.

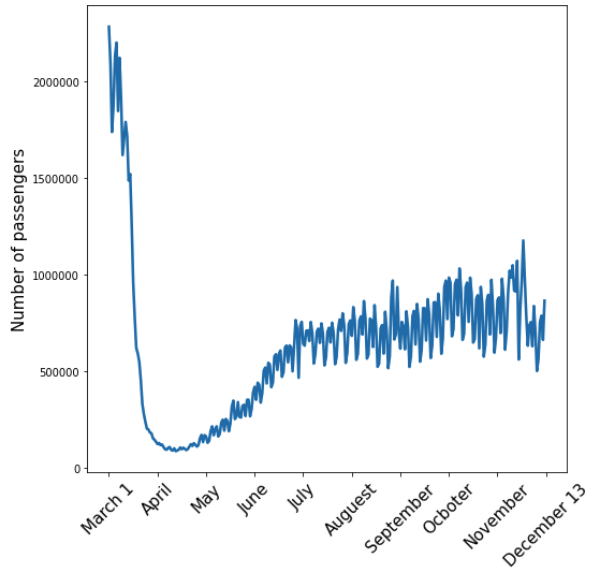
In [1]:
```python
# data from: https://www.tsa.gov/coronavirus/passenger-throughput?page=0
import pandas as pd

data=pd.read_csv("tsa_traffic_data.csv")
data.head()
```

Out[1]:

	Date	Number of passengers
0	01-Mar	2280522
1	02-Mar	2089641
2	03-Mar	1736393
3	04-Mar	1877401
4	05-Mar	2130015

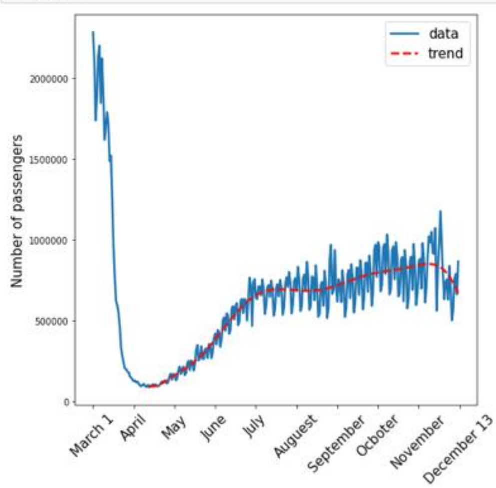
Plotting the data.

```
In [2]: import numpy as np
        import matplotlib.pyplot as plt

        label=["March 1","April","May","June","July","Auguest",
               "September","Ocboter","November","December 13"]
        plt.figure(figsize=(8,8))
        plt.plot(data['Date'],data['Number of passengers'],linewidth=2.5)
        plt.xticks(np.arange(0,320,32),labels=label,rotation=45,fontsize=15)
        plt.ticklabel_format(style='plain', axis='y')
        plt.ylabel("Number of passengers",fontsize=15)
        plt.show()
```



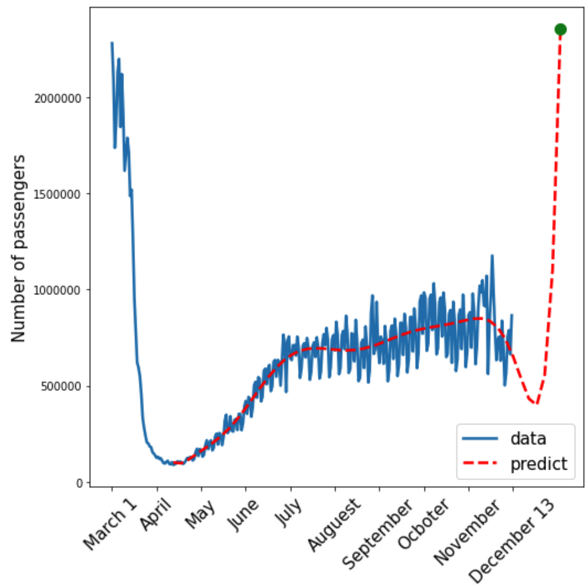
Drawing a trend line started from the minimum number of passengers by polyfit( ) from numpy library. The degree of the fitting polynomial is set as 10.

```
In [3]: start_point=data["Number of passengers"].idxmin()
        n=len(data['Date'])
        y=data["Number of passengers"][start_point:n]
        x=np.arange(start_point,n)

        trendpoly=np.poly1d(np.polyfit(x,y,10))

        plt.figure(figsize=(8,8))
        plt.plot(data['Date'],data['Number of passengers'],linewidth=2.5,label='data')
        plt.xticks(np.arange(0,320,32),labels=label,rotation=45,fontsize=15)
        plt.ticklabel_format(style='plain', axis='y')
        plt.plot(x,trendpoly(x),'--r',label='trend',linewidth=2.5)
        plt.legend(fontsize=15)
        plt.ylabel("Number of passengers",fontsize=15)
        plt.show()
```



Extending the trend line to see the estimated future number of passengers. The trend line goes down first and then keeps go up. Based on the trend, a green dot on the trend line is the time when number of passengers is recovered to the former level when there is no effect from Covid-19.

```
In [4]: x_new=np.linspace(start_point,n+34)

        plt.figure(figsize=(8,8))
        plt.plot(data['Date'],data['Number of passengers'],linewidth=2.5,label='data')
        plt.xticks(np.arange(0,320,32),labels=label,rotation=45,fontsize=15)
        plt.ticklabel_format(style='plain', axis='y')
        plt.plot(x_new,trendpoly(x_new),'--r',label='predict',linewidth=2.5)

        date=322
        plt.plot(date,trendpoly(date),'go',markersize=10)
        plt.legend(fontsize=15)
        plt.ylabel("Number of passengers",fontsize=15)
        plt.show()
```



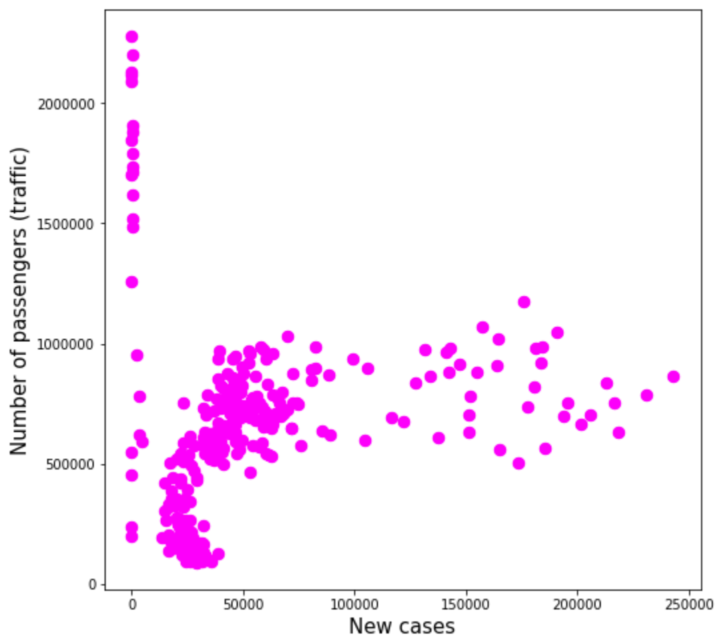
The number of passengers at the green point is 2,354,389 which is the first time over the number of passengers just before the traffic affected by Covid-19. The trend line shows taking 34 days to recover the previous traffic by subtracting the overall days from the time at the green point.

```
In [5]: print("the daily traffic just before Covid-19:",data['Number of passengers'][0])
        print(trendpoly(date))
        print(f"Find number of days to recover traffic just before Covid-19: {date-len(data['Date'])} days")

        the daily traffic just before Covid-19: 2280522
        2354389.6026456878
        Find number of days to recover traffic just before Covid-19: 34 days
```

-Correlation between Number of New Cases and Airport Traffic-

Plotting scatter points of number of passengers (y) versus new cases (x). The scatter plot below shows no noticeable correlation between two data. The number of passengers were not decreased even if more new cases reported and vice versa.

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

df=pd.read_excel("us_cases.xlsx",sheet_name='Data')
n=len(df["Unnamed: 2"])
y=df["Unnamed: 2"][4:n]
newcases_data=y[41:n]

data=pd.read_csv("tsa_traffic_data.csv")
traffic_data=data['Number of passengers']

plt.figure(figsize=(8,8))
plt.scatter(newcases_data,traffic_data,c='magenta',s=70)
plt.xlabel('New cases',fontsize=15)
plt.ylabel('Number of passengers (traffic)',fontsize=15)
plt.ticklabel_format(style='plain', axis='y')
plt.show()
```



-Features Importances for Passenger Satisfaction -

Data is from Kaggle website (https://www.kaggle.com/teejmahal20/airline-passenger-satisfaction?select=train.csv). The data is a passenger satisfaction survey from an anonymous US airline and is divided into train set and test set already. There are 24 parameters such as gender, passenger type, age, distance, and so on contributing to passenger satisfaction, and the last column contains data whether passengers satisfied their airlines or not.

Importing two data csv files named “train.csv” and “test.csv” by pandas library. “train.csv” is assigned to a variable named “train” and “test.csv” to a variable named “test” to be used for machine learning later. Taking a glance at the data.

```
In [1]: import pandas as pd
        train=pd.read_csv("train.csv")
        test=pd.read_csv("test.csv")
        train.head()
```

Out[1]:

	Unnamed: 0	id	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	...	Inflight entertainment	On-board service	Leg room service	Baggage handling	Checkin service
0	0	70172	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3	4	...	5	4	3	4	4
1	1	5047	Male	disloyal Customer	25	Business travel	Business	235	3	2	...	1	1	5	3	1
2	2	110028	Female	Loyal Customer	26	Business travel	Business	1142	2	2	...	5	4	3	4	4
3	3	24026	Female	Loyal Customer	25	Business travel	Business	562	2	5	...	2	2	5	3	1
4	4	119299	Male	Loyal Customer	61	Business travel	Business	214	3	3	...	3	3	4	4	3

Getting more details by info( ).

```
In [3]: train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103904 entries, 0 to 103903
Data columns (total 25 columns):
 #   Column                             Non-Null Count   Dtype
---  ------                             --------------   -----
 0   Unnamed: 0                         103904 non-null  int64
 1   id                                 103904 non-null  int64
 2   Gender                             103904 non-null  object
 3   Customer Type                      103904 non-null  object
 4   Age                                103904 non-null  int64
 5   Type of Travel                     103904 non-null  object
 6   Class                              103904 non-null  object
 7   Flight Distance                    103904 non-null  int64
 8   Inflight wifi service              103904 non-null  int64
 9   Departure/Arrival time convenient  103904 non-null  int64
 10  Ease of Online booking             103904 non-null  int64
 11  Gate location                      103904 non-null  int64
 12  Food and drink                     103904 non-null  int64
 13  Online boarding                    103904 non-null  int64
 14  Seat comfort                       103904 non-null  int64
 15  Inflight entertainment             103904 non-null  int64
 16  On-board service                   103904 non-null  int64
 17  Leg room service                   103904 non-null  int64
 18  Baggage handling                   103904 non-null  int64
 19  Checkin service                    103904 non-null  int64
 20  Inflight service                   103904 non-null  int64
 21  Cleanliness                        103904 non-null  int64
 22  Departure Delay in Minutes         103904 non-null  int64
 23  Arrival Delay in Minutes           103594 non-null  float64
 24  satisfaction                       103904 non-null  object
dtypes: float64(1), int64(19), object(5)
memory usage: 19.8+ MB
```

From the 25 parameters, 'Unnamed 0' parameter which is data of indices, and 'id' which is data of IDs of each passenger are unnecessary since they do not affect passenger satisfaction at all. So deleting the two parameters from both data files by drop( ). The argument axis=1 means dropping labels from the columns.

```
In [4]: train=train.drop(['Unnamed: 0','id'],axis=1)
        test=test.drop(['Unnamed: 0','id'],axis=1)
```

Finding the number of null data by searching the places by isnull( ) pandas function and adding up the number of null data by sum( ). There are 310 null data on the parameter of "Arrival Delay in Minutes" in the train data set, and 83 null data in the test data set.

```
In [5]: train.isnull().sum()

Out[5]: Gender                                 0
        Customer Type                          0
        Age                                    0
        Type of Travel                         0
        Class                                  0
        Flight Distance                        0
        Inflight wifi service                  0
        Departure/Arrival time convenient      0
        Ease of Online booking                 0
        Gate location                          0
        Food and drink                         0
        Online boarding                        0
        Seat comfort                           0
        Inflight entertainment                 0
        On-board service                       0
        Leg room service                       0
        Baggage handling                       0
        Checkin service                        0
        Inflight service                       0
        Cleanliness                            0
        Departure Delay in Minutes             0
        Arrival Delay in Minutes             310
        satisfaction                           0
        dtype: int64
```

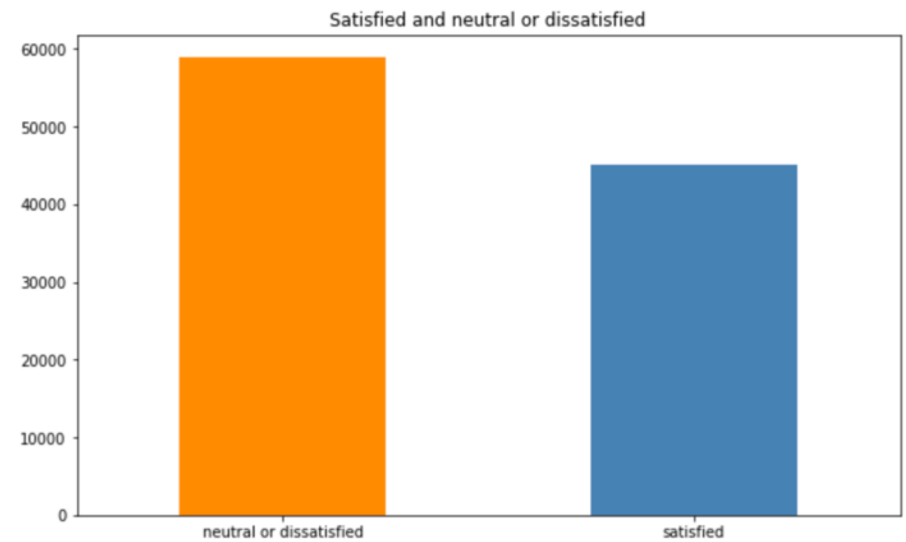
```
In [6]: test.isnull().sum()

Out[6]: Gender                                 0
        Customer Type                          0
        Age                                    0
        Type of Travel                         0
        Class                                  0
        Flight Distance                        0
        Inflight wifi service                  0
        Departure/Arrival time convenient      0
        Ease of Online booking                 0
        Gate location                          0
        Food and drink                         0
        Online boarding                        0
        Seat comfort                           0
        Inflight entertainment                 0
        On-board service                       0
        Leg room service                       0
        Baggage handling                       0
        Checkin service                        0
        Inflight service                       0
        Cleanliness                            0
        Departure Delay in Minutes             0
        Arrival Delay in Minutes              83
        satisfaction                           0
        dtype: int64
```

To replace the missing data with the mean, firstly, numpy library and sklearn library are imported. np.nan detects the NaN data and SimpleImputer from sklearn fills the empty data in with the mean. The two train and test data are put in data_list to fill with the mean for each data in a loop.

```
In [7]: import numpy as np
        from sklearn.impute import SimpleImputer

        data_list=[train,test]

        imp = SimpleImputer(missing_values=np.nan, strategy='mean')
        for dt in data_list:
            imp = imp.fit(dt[['Arrival Delay in Minutes']])
            dt['Arrival Delay in Minutes'] = imp.transform(dt[['Arrival Delay in Minutes']])
        train.isnull().sum()

Out[7]: Gender                               0
        Customer Type                        0
        Age                                  0
        Type of Travel                       0
        Class                                0
        Flight Distance                      0
        Inflight wifi service                0
        Departure/Arrival time convenient    0
        Ease of Online booking               0
        Gate location                        0
        Food and drink                       0
        Online boarding                      0
        Seat comfort                         0
        Inflight entertainment               0
        On-board service                     0
        Leg room service                     0
        Baggage handling                     0
        Checkin service                      0
        Inflight service                     0
        Cleanliness                          0
        Departure Delay in Minutes           0
        Arrival Delay in Minutes             0
        satisfaction                         0
        dtype: int64
```

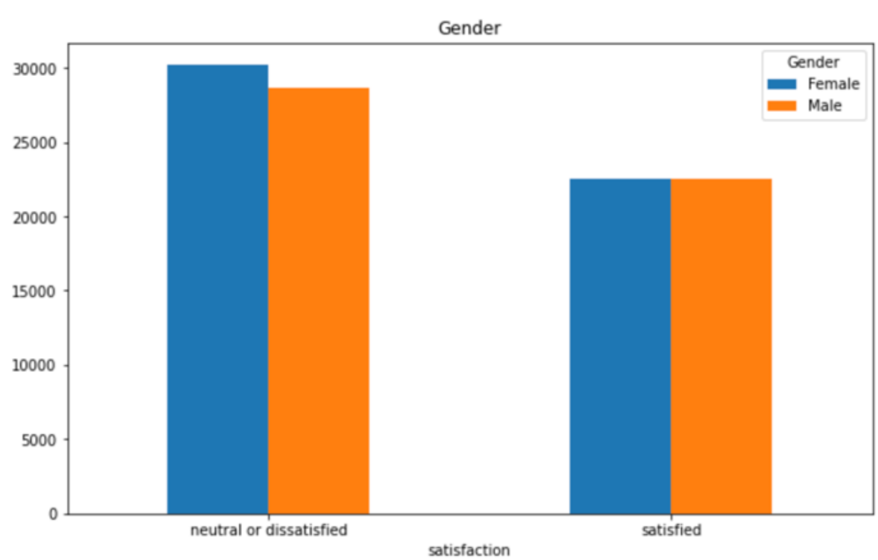
Analyzing the survey result by some parameters consisted with string categorical data in bar graphs. The graph below shows quite balanced data between satisfied and neutral or dissatisfied enough to use the train data set for machine learning.

```
In [7]: import matplotlib.pyplot as plt

        plt.figure(figsize = (10,6))
        train.satisfaction.value_counts().plot(kind='bar', color= ['darkorange','steelblue'], rot=0)
        plt.title('Satisfied and neutral or dissatisfied')
        plt.show()
```



The gender-wise distribution is quite same. However, both of male and female answered neutral or dissatisfied more than satisfied.

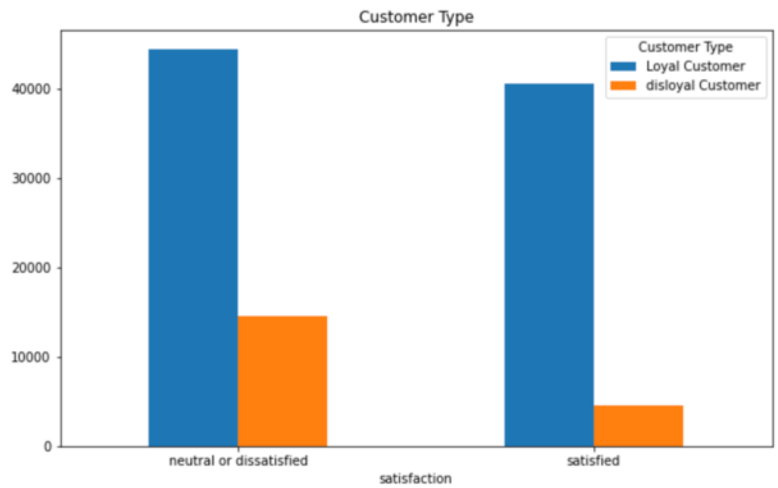
```
In [8]: class_df = train.groupby(['satisfaction','Gender'])['satisfaction'].count().unstack('Gender')
        class_df.plot(figsize=(10,6),title='Gender',kind='bar',rot=0)

Out[8]: <AxesSubplot:title={'center':'Gender'}, xlabel='satisfaction'>
```



There are much higher number of loyal customer than that of disloyal customer. Distribution is quite balanced but both of them answered neutral or dissatisfied more than satisfied.

```
In [9]: class_df = train.groupby(['satisfaction','Customer Type'])['satisfaction'].count().unstack('Customer Type')
        class_df.plot(figsize=(10,6),title='Customer Type',kind='bar',rot=0)
```

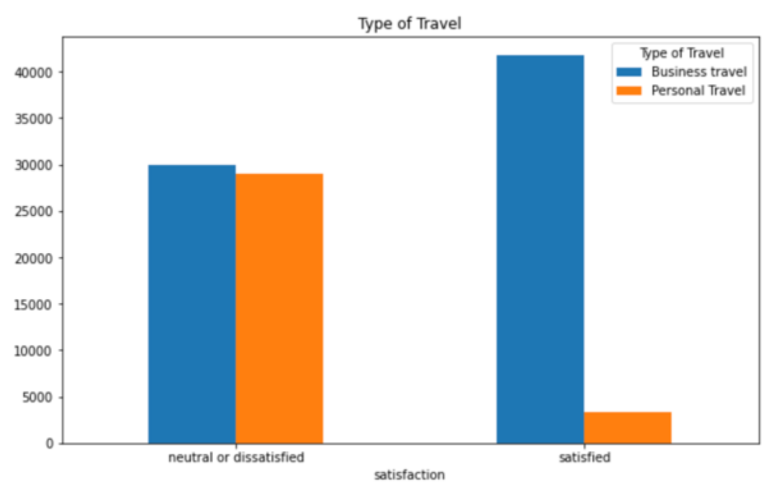
```
Out[9]: <AxesSubplot:title={'center':'Customer Type'}, xlabel='satisfaction'>
```



There are more number of business travel than that of personal travel. Interesting point is that number of satisfied passengers with business travel is higher than number of dissatisfied passengers. On the other hand, number of satisfied passengers with personal travel is much lower than number of dissatisfied passengers.

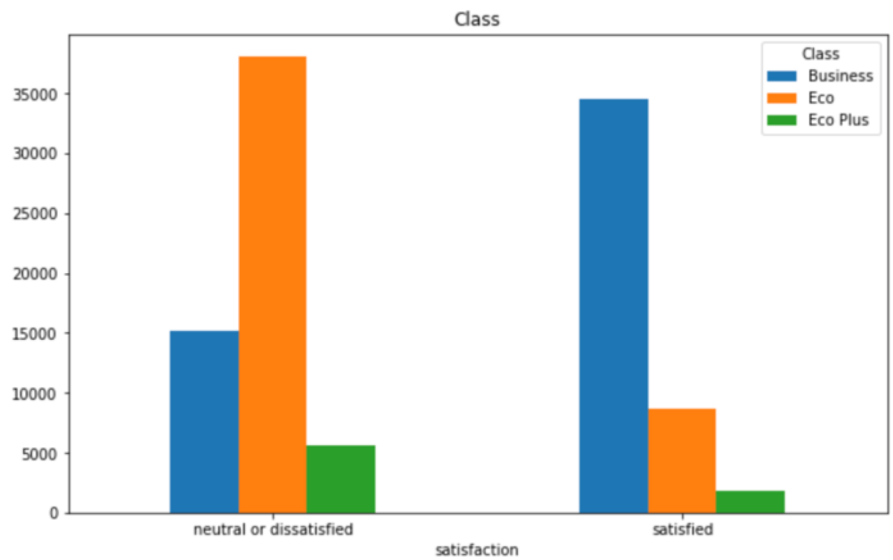
```
In [10]: class_df = train.groupby(['satisfaction','Type of Travel'])['satisfaction'].count().unstack('Type of Travel')
         class_df.plot(figsize=(10,6),title='Type of Travel',kind='bar',rot=0)
```

```
Out[10]: <AxesSubplot:title={'center':'Type of Travel'}, xlabel='satisfaction'>
```



There are three types of class; economy, economy plus, and business. Number of passengers using economy satisfied is higher than that dissatisfied. Number of passengers using economy plus satisfied is lower than that dissatisfied. Only number of passengers using business satisfied is higher than that dissatisfied.

```
In [11]: class_df = train.groupby(['satisfaction','Class'])['satisfaction'].count().unstack('Class')
         class_df.plot(figsize=(10,6),title='Class',kind='bar',rot=0)

Out[11]: <AxesSubplot:title={'center':'Class'}, xlabel='satisfaction'>
```

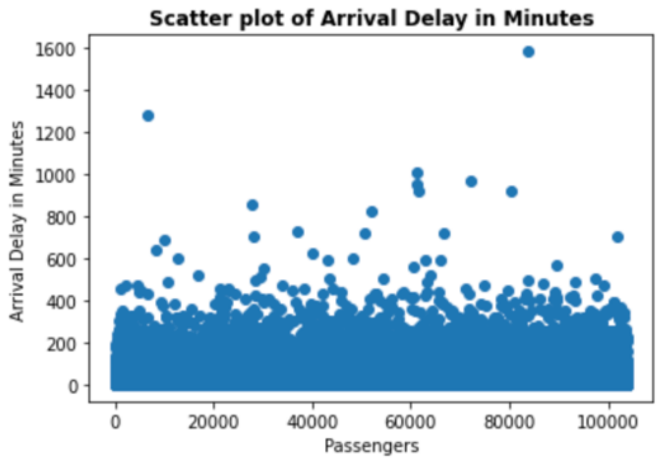


From sklearn.preprocessing, LabelEncoder is imported to covert string data into integer data in order to execute machine learning models. The two data sets in data_list are changed in a loop which lists column names and coverts by fit_transform( ) function if the type of the first element of the column is string.

```
In [8]: from sklearn.preprocessing import LabelEncoder
        le = LabelEncoder()
        for dt in data_list:
          for i in dt.columns.values.tolist():
            if type(dt[i][0])==str:
              dt[i] = le.fit_transform(dt[i])
        train.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 103904 entries, 0 to 103903
        Data columns (total 23 columns):
         #   Column                             Non-Null Count   Dtype
        ---  ------                             --------------   -----
         0   Gender                             103904 non-null  int64
         1   Customer Type                      103904 non-null  int64
         2   Age                                103904 non-null  int64
         3   Type of Travel                     103904 non-null  int64
         4   Class                              103904 non-null  int64
         5   Flight Distance                    103904 non-null  int64
         6   Inflight wifi service              103904 non-null  int64
         7   Departure/Arrival time convenient  103904 non-null  int64
         8   Ease of Online booking             103904 non-null  int64
         9   Gate location                      103904 non-null  int64
         10  Food and drink                     103904 non-null  int64
         11  Online boarding                    103904 non-null  int64
         12  Seat comfort                       103904 non-null  int64
         13  Inflight entertainment             103904 non-null  int64
         14  On-board service                   103904 non-null  int64
         15  Leg room service                   103904 non-null  int64
         16  Baggage handling                   103904 non-null  int64
         17  Checkin service                    103904 non-null  int64
         18  Inflight service                   103904 non-null  int64
         19  Cleanliness                        103904 non-null  int64
         20  Departure Delay in Minutes         103904 non-null  int64
         21  Arrival Delay in Minutes           103904 non-null  float64
         22  satisfaction                       103904 non-null  int64
        dtypes: float64(1), int64(22)
        memory usage: 18.2 MB
```

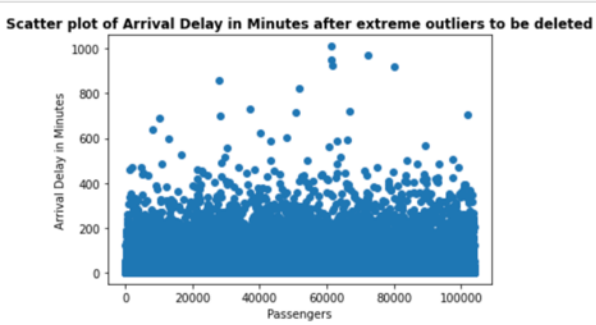
After filling all the null data with the mean, it is necessary to delete outliers which would lower an accuracy of a machine learning model. To visualize the 'Arrival Delay in Minutes' data in a form of scatter plot, matplotlib library. From the scatter plots below, there are two conspicuous outliers over 1200 minutes.

```
In [9]: import matplotlib.pyplot as plt

        plt.figure()
        xx=np.linspace(0,len(train),len(train))
        plt.scatter(xx, train['Arrival Delay in Minutes'])
        plt.xlabel('Passengers')
        plt.ylabel('Arrival Delay in Minutes')
        plt.title('Scatter plot of Arrival Delay in Minutes',fontweight='bold')
        plt.show()
```



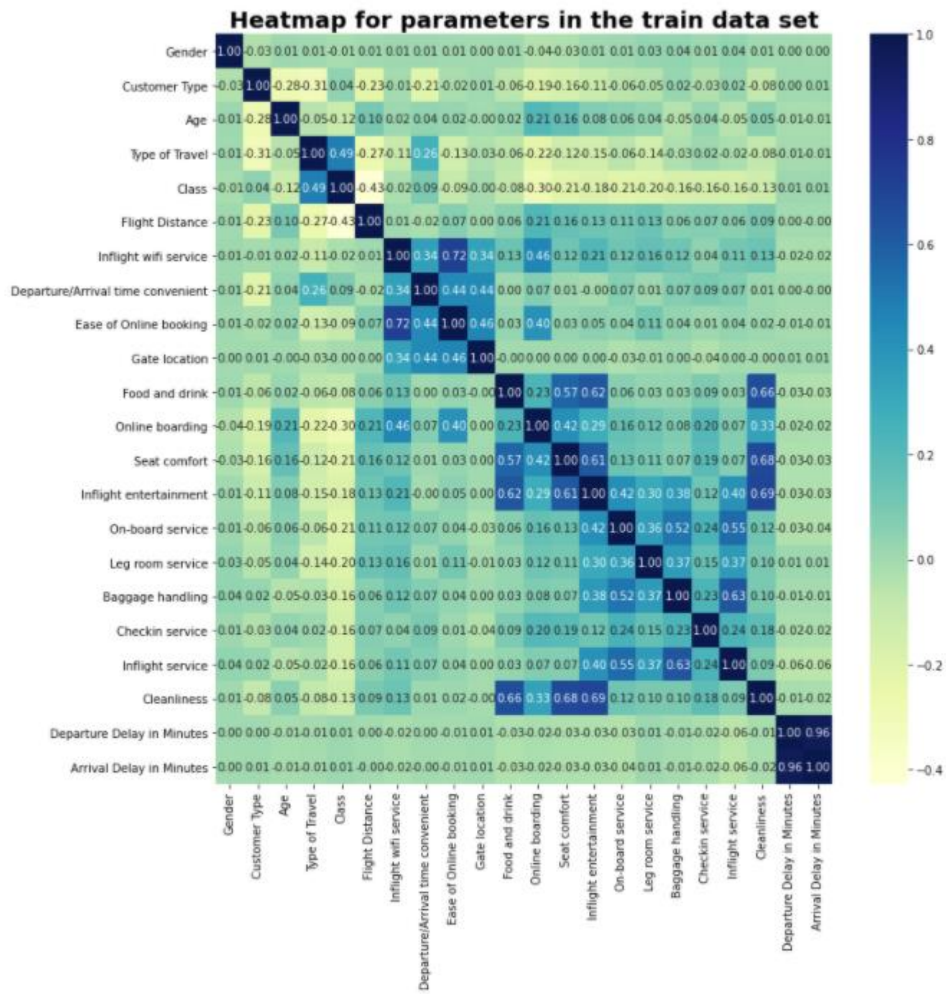
Deleting the two outliers and checking they are deleted completely by plotting the new data.

```
In [10]: train = train[train['Arrival Delay in Minutes'] <= 1200]

         plt.figure()
         xx=np.linspace(0,len(train),len(train))
         plt.scatter(xx, train['Arrival Delay in Minutes'])
         plt.xlabel('Passengers')
         plt.ylabel('Arrival Delay in Minutes')
         plt.title('Scatter plot of Arrival Delay in Minutes after extreme outliers to be deleted',fontweight='bold')
         plt.show()
```



Creating a heatmap to visualize relationships between one by one parameter makes easier to see data correlations at a glance. Seaborn library is used for drawing the heatmap. From the heatmap below, for example, the correlation between ‘Ease of Online booking’ and ‘Inflight WIFI service’ shows blue block (0.72), which means those have highly positive correlation.

In [15]:
```
import seaborn as sns
plt.figure(figsize=(12,12))
list_columns=train.drop('satisfaction',axis=1).columns.values.tolist()
sns.heatmap(train[list_columns].corr(),annot=True,fmt=".2f",cmap='YlGnBu')
plt.title('Heatmap for parameters in the train data set',fontweight='bold',fontsize=20)
# plt.xticks(rotation=70)
plt.show()
```



Ensemble models from sklearn library can be implemented with the pre-processed data. There are various ensemble models provided by sklearn, divided into two groups: classification model and regression model. Classification deals with multiple categorical classes or discrete values, otherwise regression deals with continuous real values. Since the data sets used in this research are discrete values, classification models would perform better than regression models. 6 models are used to confirm it. The train and test data sets are separated into input variables (X) and output variable (Y). Setting identical n_estimators, max_depth, min_samples_split, random_state for all the models.

```
In [13]: # #dataset
         y_train=train["satisfaction"]
         x_train=train.drop(["satisfaction"],axis=1)
         y_test=test["satisfaction"]
         x_test=test.drop(["satisfaction"],axis=1)
```

```
In [14]: from sklearn.ensemble import RandomForestClassifier
         clf=RandomForestClassifier(n_estimators=100,max_depth=None,min_samples_split=2,random_state=8)
         clf.fit(x_train,y_train)
         acc_rfc= (clf.score(x_test,y_test),'RandomForestClassifier')
         print(f"Accuracy of RandomForestClassifier: {round(acc_rfc[0],3)}")

         Accuracy of RandomForestClassifier: 0.963
```

```
In [15]: from sklearn.ensemble import RandomForestRegressor
         clf2=RandomForestRegressor(n_estimators=100,max_depth=None,min_samples_split=2,random_state=8)
         clf2.fit(x_train,y_train)
         acc_rfr= (clf2.score(x_test,y_test),'RandomForestRegressor')
         print(f"Accuracy of RandomForestRegressor: {round(acc_rfr[0],3)}")

         Accuracy of RandomForestRegressor: 0.886
```

```
In [16]: from sklearn.ensemble import ExtraTreesClassifier
         clf3=ExtraTreesClassifier(n_estimators=100,max_depth=None,min_samples_split=2,random_state=8)
         clf3.fit(x_train,y_train)
         acc_etc = (clf3.score(x_test,y_test),'ExtraTreesClassifier')
         print(f"Accuracy of ExtraTreeClassifier: {round(acc_etc[0],3)}")

         Accuracy of ExtraTreeClassifier: 0.963
```

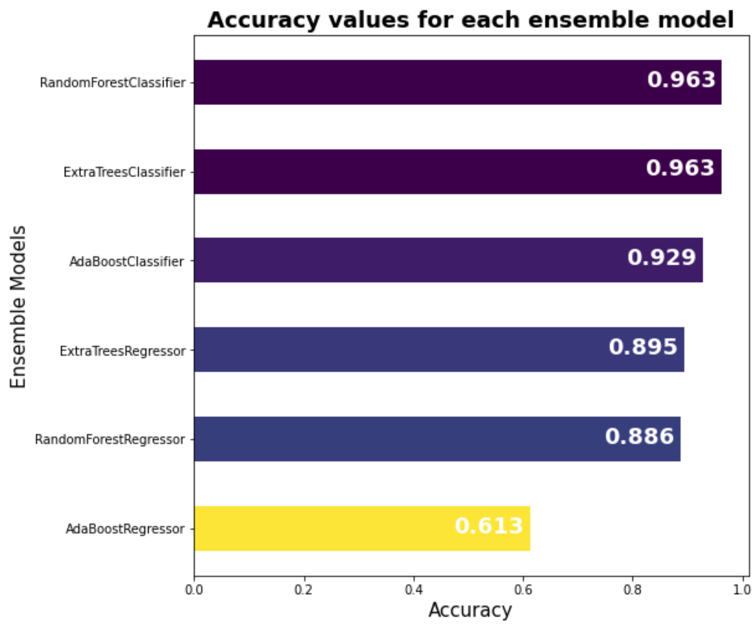
```
In [17]: from sklearn.ensemble import ExtraTreesRegressor
         clf4=ExtraTreesRegressor(n_estimators=100,max_depth=None,min_samples_split=2,random_state=8)
         clf4.fit(x_train,y_train)
         acc_etr = (clf4.score(x_test,y_test),'ExtraTreesRegressor')
         print(f"Accuracy of ExtraTreeRegressor: {round(acc_etr[0],3)}")

         Accuracy of ExtraTreeRegressor: 0.895
```

```
In [18]: from sklearn.ensemble import AdaBoostClassifier
         clf5=AdaBoostClassifier(n_estimators=100,random_state=8)
         clf5.fit(x_train,y_train)
         acc_abc = (clf5.score(x_test,y_test),'AdaBoostClassifier')
         print(f"Accuracy of AdaBoostClassifier: {round(acc_abc[0],3)}")

         Accuracy of AdaBoostClassifier: 0.929
```

```
In [19]: from sklearn.ensemble import AdaBoostRegressor
         clf6=AdaBoostRegressor(n_estimators=100,random_state=8)
         clf6.fit(x_train,y_train)
         acc_abr = (clf6.score(x_test,y_test),'AdaBoostRegressor')
         print(f"Accuracy of AdaBoostRegressor: {round(acc_abr[0],3)}")

         Accuracy of AdaBoostRegressor: 0.613
```

Making a horizontal bar chart for easier comparison among the accuracy values. The accuracy values are put in all_models list and sorted from the highest to lowest value by sorted( ) function. The chart below shows classification models perform better than regression models as expected, and both of RandomForestClassifier and ExtraTreesClassifier have the highest accuracy value, 0.963.

```
In [23]: all_models=[acc_rfc,acc_rfr,acc_etc,acc_etr,acc_abc,acc_abr]
         all_models=sorted(all_models,key=lambda x:x[0])

         accuracy_values=[x[0] for x in all_models]
         models=[x[1] for x in all_models]

         plt.figure(figsize=(8,8))
         my_cmap = plt.get_cmap("viridis").reversed()
         rescale = lambda y: (y-np.min(y)) / (np.max(y) - np.min(y))
         plt.barh(models,accuracy_values,height=0.5,align='center', color=my_cmap(rescale(accuracy_values)))
         for i, v in enumerate(accuracy_values):
             plt.text(v-0.14,i-.05,str(round(v,3)),color='white',fontsize=18,fontweight='bold')
         plt.ylabel('Ensemble Models',fontsize=15)
         plt.xlabel('Accuracy',fontsize=15)
         plt.title('Accuracy values for each ensemble model',fontsize=18,fontweight='bold')
         plt.show()
```



Having the highest accuracy, RandomForestClassifier model is selected to find factors from most to least importance affecting passenger satisfaction. Before doing so, raising the accuracy by searching the best parameter values for an estimator. GridSearchCV built in sklearn is used.

```
In [24]: from sklearn.model_selection import GridSearchCV
param_grid={
    'n_estimators':[150,200,250,300],
    'max_depth':[None,6],
    'min_samples_split':[0.1,2],
    'max_features':['auto','sqrt','log2'],
    'random_state':[None,1,3,5,7,9],
}

estimator=RandomForestClassifier()

grid_search = GridSearchCV(estimator=estimator,param_grid=param_grid,n_jobs=-1)
grid_search.fit(x_train,y_train)

print(grid_search.best_params_)
```

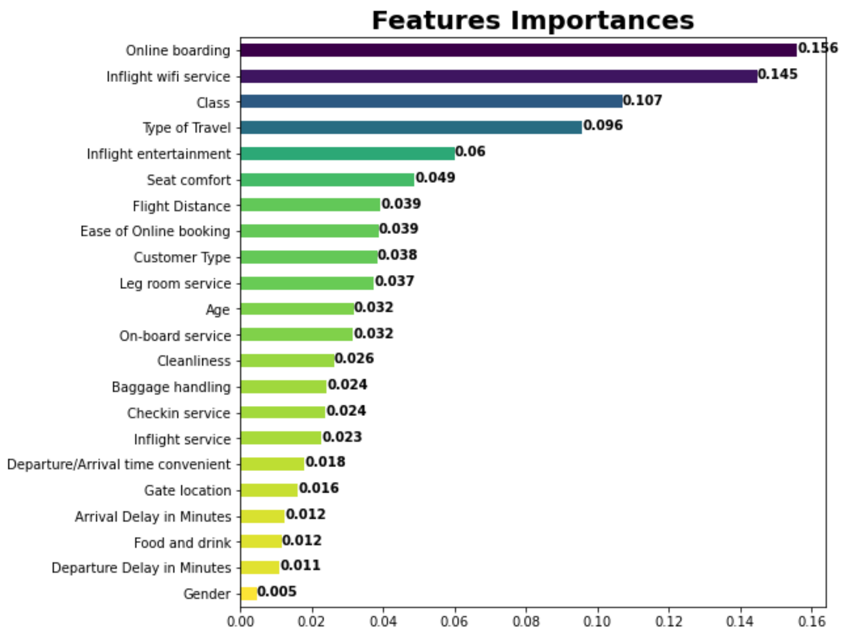
Feature importance for each parameter to passenger satisfaction can be calculated by feature_importances_ built in sklearn. The higher, the more important and influential the feature. The accuracy increases from 0.963 to 0.964 through GridSearchCV.

```
In [27]: clf_best=RandomForestClassifier(n_estimators=250,max_depth=None,min_samples_split=2,max_features='sqrt',random_state=3)
clf_best.fit(x_train,y_train)
print(f"Accuracy of new RandomForestClassifier: {clf_best.score(x_test,y_test)}\n")
dic=dict(zip(x_train.columns,clf_best.feature_importances_))
for item in sorted(dic.items(), key=lambda x: x[1], reverse=True):
    print(item[0],round(item[1],4))
```

```
Accuracy of new RandomForestClassifier: 0.9636202648598706

Online boarding 0.156
Inflight wifi service 0.1448
Class 0.1069
Type of Travel 0.0957
Inflight entertainment 0.06
Seat comfort 0.0488
Flight Distance 0.0393
Ease of Online booking 0.0387
Customer Type 0.0383
Leg room service 0.0374
Age 0.0318
On-board service 0.0315
Cleanliness 0.0263
Baggage handling 0.0243
Checkin service 0.0238
Inflight service 0.0227
Departure/Arrival time convenient 0.018
Gate location 0.0161
Arrival Delay in Minutes 0.0124
Food and drink 0.0115
Departure Delay in Minutes 0.0111
Gender 0.0046
```

Visualizing the result in a horizontal bar chart. As sorting the values from the highest, online boarding is the most important parameter and gender is the least.

```
In [29]: importances_sorted = pd.Series(data=clf_best.feature_importances_,index= x_train.columns).sort_values()
plt.figure(figsize=(8,8))
importances_sorted.plot(kind='barh',color=my_cmap(rescale(importances_sorted)))
for i, v in enumerate(importances_sorted):
    plt.text(v,i-.1,str(round(v,3)),color='black',fontweight='bold')
plt.title('Features Importances',fontweight='bold',fontsize=20)
plt.show()
```



The traffic trend calculated by polyfit( ) and poly1d( ) functions shows the trend line keeps go up and recovers the traffic before affected by Covid-19 from current situation in only 34 days. The increasing trend is reliable to some extent, but such the rapid increasing rate is suspicious therefore it would be better to use other prediction techniques related with deep learning in order to predict the future traffic more precisely.

From the trends of number of new cases and passengers calculated by numpy, the traffic would increase regardless of increasing new cases for now. The increasing rate of the traffic would even more rapidly rise if vaccine efficacy works in the US where has started to provide the vaccine from December 14. Therefore, airline companies have to improve their services as soon as possible to prepare for attracting potential passengers.

The priority of upgrading their services is from the highest feature importance for passenger satisfaction calculated by randomforestclassifier machine learning model which has the highest accuracy: ‘online boarding’ (0.156). It is followed by ‘inflight WIFI service’ (0.145), ‘class’ (0.107), ‘type of Travel’ (0.096), ‘inflight entertainment’ (0.06), ‘seat comfort’ (0.049), and so on. Especially, there are many numbers of dissatisfied passengers who used economy or economy plus service in ‘class’ and their flights for personal travel purposes in ‘type of Travel’ compared to the other groups in the same parameter. Those particular inconveniences should be improved in the respective services. Thus, the airline companies in the US ought to provide better service in order to satisfy more passengers and not fall behind in the competition of attracting as many potential passengers as possible from now on by improving in order from the highest feature importance, online boarding service.