FLIGHT PRICES DETECTION

INTRODUCTION

- The objective of the study is to analyse the flight booking dataset obtained from “Ease My Trip” website and to conduct various statistical hypothesis tests in order to get meaningful information from it.
- The 'Linear Regression' statistical algorithm would be used to train the dataset and predict a continuous target variable.
- 'Easemytrip' is an internet platform for booking flight tickets, and hence a platform that potential passengers use to buy tickets.
- A thorough study of the data will aid in the discovery of valuable insights that will be of enormous value to passengers.

DATA COLLECTION AND METHODOLOGY:

- Octoparse scraping tool was used to extract data from the website.
- Data was collected in two parts: one for economy class tickets and another for business class tickets.
- A total of 300261 distinct flight booking options was extracted from the site.
- Data was collected for 50 days, from February 11th to March 31st, 2022.
- Data source was secondary data and was collected from Ease my trip website.

DATASET

- Dataset contains information about flight booking options from the website Easemytrip for flight travel between India's top 6 metro cities.
- There are 300261 datapoints and 11 features in the cleaned dataset.

NUMPY:

- NumPy stands for Numerical Python & also known by the alias array .
- NumPy is a Python library used for working with arrays.
- It also has functions for working in domain of linear algebra, fourier transform, and matrices.
- It is an open source project and you can use it freely.
- It's the universal standard for working with numerical data in Python, At the core of the scientific Python & PyData ecosystems.
- NumPy functions are used to create, manipulate, and analyze NumPy arrays.
- The syntax of NumPy functions generally involves calling the function and passing one or more parameters, such as the shape of the array, the range of values to generate or the type of data to use.

- The list of most commonly used numeric data types in NumPy: int8 , int16 , int32 , int64 - signed integer types with different bit sizes. uint8 , uint16 , uint32 , uint64 - unsigned integer types with different bit sizes.

PANDAS:

- Pandas is a software library written for the Python programming language for functions: analyzing, cleaning, exploring & manipulating data.
- Also used for working with data sets.
- The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis".
- In particular, it offers data structures and operations for manipulating numerical tables and time series.
- It is free software released under the three-clause BSD license.
- Pandas is built on top of the NumPy package, meaning a lot of the structure of NumPy is used or replicated in Pandas.
- Data in pandas is often used to feed statistical analysis in SciPy, plotting functions from Matplotlib & ML algorithms in Scikit-learn.
- It provides many functions & methods to expedite the data analysis process.
- What makes pandas so common is its functionality, flexibility & simple syntax.

MATPLOTLIB.PYPLOT:

- matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB.
- Matplotlib is a powerful library which is used to visualize data in form of plots.
- We'll see that, with the help of this library one can create many types of plots like line, bar, histogram, contour, scatter, violin, and many more.
- It is used for creating 2D plots of Arrays.
- Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.
- The Matplotlib architecture is composed of three main layers: 1.Backend Layer — Handles all the heavy works via communicating to the drawing toolkits in your machine. It is the most complex layer. 2.Artist Layer — Allows full control and fine-tuning of the Matplotlib figure — the top-level container for all plot elements.

SEABORN:

- Seaborn is a Python data visualization library based on matplotlib.
- Python Seaborn library is a widely popular data visualization library that is commonly used for data science and machine learning tasks.
- We'll build it on top of the matplotlib data visualization library and can perform exploratory analysis.
- It provides a high-level interface for drawing attractive and informative statistical graphics.
- For a brief introduction to the ideas behind the library, you can read the introductory notes.

- Provides five different built-in themes named: "darkgrid", "whitegrid", "dark", "white" & "ticks". We can select them both with the style argument of the set_style or set_style function.

What is the difference between matplotlib and seaborn?

- Matplotlib is a library in Python that enables users to generate visualizations like histograms, scatter plots, bar charts, pie charts and much more.
- Seaborn is a visualization library that is built on top of Matplotlib & It provides data visualizations that are typically more aesthetic

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

IMPORTING & READING THE EXCEL FILE

In [2]:
```
df = pd.read_csv(r'E:\DATASETS/flight_price.csv',sep=',')
df
```

Out[2]:

	airline	flight	source_city	departure_time	stops	arrival_time	destination_city	class	duration	days_left	price
0	SpiceJet	SG-8709	Delhi	Evening	zero	Night	Mumbai	Economy	2.17	1	5953
1	SpiceJet	SG-8157	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.33	1	5953
2	AirAsia	I5-764	Delhi	Early_Morning	zero	Early_Morning	Mumbai	Economy	2.17	1	5956
3	Vistara	UK-995	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.25	1	5955
4	Vistara	UK-963	Delhi	Morning	zero	Morning	Mumbai	Economy	2.33	1	5955
...	...	...	...	...	...	...	...	...	...	...	...
300148	Vistara	UK-822	Chennai	Morning	one	Evening	Hyderabad	Business	10.08	49	69265
300149	Vistara	UK-826	Chennai	Afternoon	one	Night	Hyderabad	Business	10.42	49	77105
300150	Vistara	UK-832	Chennai	Early_Morning	one	Night	Hyderabad	Business	13.83	49	79099
300151	Vistara	UK-828	Chennai	Early_Morning	one	Evening	Hyderabad	Business	10.00	49	81585
300152	Vistara	UK-822	Chennai	Morning	one	Evening	Hyderabad	Business	10.08	49	81585

300153 rows × 11 columns

head() function:

- The head() function is used to get the first n rows.
- This function returns the first n rows for the object based on position.
- It is useful for quickly testing if your object has the right type of data in it.
- The first n rows of the caller object.

In [3]:
```
df.head(10)
```

Out[3]:

	airline	flight	source_city	departure_time	stops	arrival_time	destination_city	class	duration	days_left	price
0	SpiceJet	SG-8709	Delhi	Evening	zero	Night	Mumbai	Economy	2.17	1	5953
1	SpiceJet	SG-8157	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.33	1	5953
2	AirAsia	I5-764	Delhi	Early_Morning	zero	Early_Morning	Mumbai	Economy	2.17	1	5956
3	Vistara	UK-995	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.25	1	5955
4	Vistara	UK-963	Delhi	Morning	zero	Morning	Mumbai	Economy	2.33	1	5955
5	Vistara	UK-945	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.33	1	5955
6	Vistara	UK-927	Delhi	Morning	zero	Morning	Mumbai	Economy	2.08	1	6060
7	Vistara	UK-951	Delhi	Afternoon	zero	Evening	Mumbai	Economy	2.17	1	6060
8	GO_FIRST	G8-334	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.17	1	5954
9	GO_FIRST	G8-336	Delhi	Afternoon	zero	Evening	Mumbai	Economy	2.25	1	5954

tail() function:

- The tail() function is used to get the last n rows.
- This function returns last n rows from the object based on position.
- It is useful for quickly verifying data, for example, after sorting or appending rows.
- The last n rows of the caller object.

In [4]:
```
df.tail(10)
```

Out[4]:

	airline	flight	source_city	departure_time	stops	arrival_time	destination_city	class	duration	days_left	price
300143	Air_India	AI-440	Chennai	Early_Morning	one	Night	Hyderabad	Business	17.42	49	51345
300144	Air_India	AI-539	Chennai	Evening	one	Morning	Hyderabad	Business	18.92	49	51345
300145	Air_India	AI-430	Chennai	Morning	one	Morning	Hyderabad	Business	23.08	49	51345
300146	Air_India	AI-440	Chennai	Early_Morning	one	Morning	Hyderabad	Business	26.83	49	51345
300147	Air_India	AI-569	Chennai	Early_Morning	one	Night	Hyderabad	Business	17.25	49	68739
300148	Vistara	UK-822	Chennai	Morning	one	Evening	Hyderabad	Business	10.08	49	69265
300149	Vistara	UK-826	Chennai	Afternoon	one	Night	Hyderabad	Business	10.42	49	77105
300150	Vistara	UK-832	Chennai	Early_Morning	one	Night	Hyderabad	Business	13.83	49	79099
300151	Vistara	UK-828	Chennai	Early_Morning	one	Evening	Hyderabad	Business	10.00	49	81585
300152	Vistara	UK-822	Chennai	Morning	one	Evening	Hyderabad	Business	10.08	49	81585

SHAPE OF THE DATAFRAME:

- The shape attribute of pandas.
- DataFrame stores the number of rows and columns as a tuple (number of rows, number of columns).
- The shape function in Python is used to find the dimensions of data structures, such as NumPy arrays and Pandas DataFrames.

In [5]:
```
df.shape
```

Out[5]:
```
(300153, 11)
```

info() function:

- The info() function is used to print a concise summary of a DataFrame.
- This method prints information about a DataFrame including the index dtype and column dtypes, non-null values and memory usage.

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300153 entries, 0 to 300152
Data columns (total 11 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   airline           300153 non-null  object 
 1   flight            300153 non-null  object 
 2   source_city       300153 non-null  object 
 3   departure_time    300153 non-null  object 
 4   stops             300153 non-null  object 
 5   arrival_time      300153 non-null  object 
 6   destination_city  300153 non-null  object 
 7   class             300153 non-null  object 
 8   duration          300153 non-null  float64
 9   days_left         300153 non-null  int64  
 10  price             300153 non-null  int64  
dtypes: float64(1), int64(2), object(8)
memory usage: 25.2+ MB
```

describe() function:

- Pandas describe() is used to view some basic statistical details like percentile, mean, std etc. of a data frame or a series of numeric values.
- When this method is applied to a series of string, it returns a different output which is shown below.

include = 'all' function:

- It provides as an option, the result will include a union of attributes of each type.
- The includes and excludes parameters, can be used to limit which columns in a DataFrame are analyzed for the output.
- The parameters are ignored when analyzing a Series & describes a numeric Series.

In [7]:
```
df.describe(include = "all")
```

Out[7]:

	airline	flight	source_city	departure_time	stops	arrival_time	destination_city	class	duration	days_left	
count	300153	300153	300153	300153	300153	300153	300153	300153	300153.000000	300153.000000	300
unique	6	1561	6	6	3	6	6	2	NaN	NaN	
top	Vistara	UK-706	Delhi	Morning	one	Night	Mumbai	Economy	NaN	NaN	
freq	127859	3235	61343	71146	250863	91538	59097	206666	NaN	NaN	
mean	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	12.221021	26.004751	20
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	7.191997	13.561004	22
min	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.830000	1.000000	1
25%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	6.830000	15.000000	4
50%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	11.250000	26.000000	7
75%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	16.170000	38.000000	42
max	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	49.830000	49.000000	123

FEATURES.

- **1. Airline:** The name of the airline company is stored in the airline column. It is a categorical feature having 6 different airlines.
- **2. Flight:** Flight stores information regarding the plane's flight code. It is a categorical feature.
- **3. Source City:** City from which the flight takes off. It is a categorical feature having 6 unique cities.
- **4. Departure Time:** This is a derived categorical feature obtained created by grouping time periods into bins. It stores information about the departure time & have 6 unique time labels.
- **5. Stops:** A categorical feature with 3 distinct values that stores the number of stops between the source and destination cities.
- **6. Arrival Time:** This is a derived categorical feature created by grouping time intervals into bins. It has six distinct time labels and keeps information about the arrival time.
- **7. Destination City:** City where the flight will land. It is a categorical feature having 6 unique cities.
- **8. Class:** A categorical feature that contains information on seat class; it has two distinct values: Business and Economy.
- **9. Duration:** A continuous feature that displays the overall amount of time it takes to travel between cities in hours.
- **10. Days Left:** This is a derived characteristic that is calculated by subtracting the trip date by the booking date.
- **11. Price:** Target variable stores information of the ticket price.

drop() function:

- The drop() function is used to drop specified labels from rows or columns.
- Remove rows or columns by specifying label names & corresponding axis or by specifying directly index or column names.
- When using a multi-index, labels on different levels can be removed by specifying the level.

Delete unnecessary column

In [8]:
```
df = df.drop(columns=['flight','days_left'])
df
```

Out[8]:

	airline	source_city	departure_time	stops	arrival_time	destination_city	class	duration	price
0	SpiceJet	Delhi	Evening	zero	Night	Mumbai	Economy	2.17	5953
1	SpiceJet	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.33	5953
2	AirAsia	Delhi	Early_Morning	zero	Early_Morning	Mumbai	Economy	2.17	5956
3	Vistara	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.25	5955
4	Vistara	Delhi	Morning	zero	Morning	Mumbai	Economy	2.33	5955
...	...	...	...	...	...	...	...	...	...
300148	Vistara	Chennai	Morning	one	Evening	Hyderabad	Business	10.08	69265
300149	Vistara	Chennai	Afternoon	one	Night	Hyderabad	Business	10.42	77105
300150	Vistara	Chennai	Early_Morning	one	Night	Hyderabad	Business	13.83	79099
300151	Vistara	Chennai	Early_Morning	one	Evening	Hyderabad	Business	10.00	81585
300152	Vistara	Chennai	Morning	one	Evening	Hyderabad	Business	10.08	81585

300153 rows × 9 columns

Label encoding:

- Label encoding is a technique used in ML & data analysis to convert categorical variables into numerical format.
- It is particularly useful when working with algorithms that require numerical input, as most ML models can only operate on numerical data.
- The basic idea of label encoding is to assign a unique integer to each category in a categorical variable.

- For example, if we have a categorical variable “colour” with categories “red”, “green”, and “blue”, we can assign the labels 0, 1 and 2 respectively.

When should I use label encoding?

- Variables that reflect distinct categories such as colors, nations or product types are known as categorical variables.
- However, since the majority of ML algorithms operate on numerical data, categorical variables must be transformed into numeric form - Label encoding can be useful in this situation.

Scikit-learn:

- An open source data analysis library and the gold standard for Machine Learning (ML) in the Python ecosystem.
- Scikit-learn is a Python module for machine learning built on top of SciPy (Scientific Python).
- Key concepts and features include: Algorithmic decision-making methods, including: Classification: identifying & categorizing data based on patterns.

What is scikit-learn used for?

- Scikit-Learn, also known as sklearn is a python library to implement machine learning models and statistical modelling.
- Through scikit-learn, we implement various ML models for regression, classification, clustering & statistical tools for analyzing these models.

- We need to do label encoding because we want to do PCA process which don't support string datatype.

In [9]:
```
from sklearn import preprocessing as pre

le = pre.LabelEncoder()
le.fit(df['airline'])
df['airline'] = le.transform(df['airline'])
airline_labels = dict(zip(le.classes_, le.transform(le.classes_)))
print(airline_labels)
```

```
{'AirAsia': 0, 'Air_India': 1, 'GO_FIRST': 2, 'Indigo': 3, 'SpiceJet': 4, 'Vistara': 5}
```

source_city

In [10]:
```
le.fit(df['source_city'])
df['source_city'] = le.transform(df['source_city'])
source_labels = dict(zip(le.classes_, le.transform(le.classes_)))
print(source_labels)
```

```
{'Bangalore': 0, 'Chennai': 1, 'Delhi': 2, 'Hyderabad': 3, 'Kolkata': 4, 'Mumbai': 5}
```

stops

In [11]:
```
le.fit(df['stops'])
df['stops'] = le.transform(df['stops'])
stops_labels = dict(zip(le.classes_, le.transform(le.classes_)))
print(stops_labels)
```

```
{'one': 0, 'two_or_more': 1, 'zero': 2}
```

departure time

In [12]:
```
le.fit(df['departure_time'])
df['departure_time'] = le.transform(df['departure_time'])
deptime_labels = dict(zip(le.classes_, le.transform(le.classes_)))
print(deptime_labels)
```

```
{'Afternoon': 0, 'Early_Morning': 1, 'Evening': 2, 'Late_Night': 3, 'Morning': 4, 'Night': 5}
```

arrival time

In [13]:
```
le.fit(df['arrival_time'])
df['arrival_time'] = le.transform(df['arrival_time'])
arrtime_labels = dict(zip(le.classes_, le.transform(le.classes_)))
print(arrtime_labels)
```

```
{'Afternoon': 0, 'Early_Morning': 1, 'Evening': 2, 'Late_Night': 3, 'Morning': 4, 'Night': 5}
```

destination city

In [14]:
```
le.fit(df['destination_city'])
df['destination_city'] = le.transform(df['destination_city'])
des_labels = dict(zip(le.classes_, le.transform(le.classes_)))
print(des_labels)
```

```
{'Bangalore': 0, 'Chennai': 1, 'Delhi': 2, 'Hyderabad': 3, 'Kolkata': 4, 'Mumbai': 5}
```

class seat

In [15]:
```
le.fit(df['class'])
df['class'] = le.transform(df['class'])
class_labels = dict(zip(le.classes_, le.transform(le.classes_)))
print(class_labels)
```

```
{'Business': 0, 'Economy': 1}
```

iloc[ ] function:

- iloc is integer location.
- It returns a Pandas Series when one row is selected, and a Pandas DataFrame when multiple rows are selected, or if any column in full is selected. (OR)
- The iloc() function in python is one of the functions defined in the Pandas module that helps us to select a specific row or column from the data set.
- To counter this, pass a single-valued list if you require DataFrame output.
- Using the iloc() function in python, we can easily retrieve any particular value from a row or column using index values.

What is the iloc operator used for?

- The iloc operator in Python is used to select and access data in a DataFrame by its integer position, allowing you to grab specific rows or columns using numerical indices.

Separating variable and target

In [16]:
```
X = df.iloc[:,:-1]
Y = df.iloc[:,1]
print(X)
print(Y)

dataframe = pd.DataFrame(df)
dataframe
```

```
        airline  source_city  departure_time  stops  arrival_time  \
0             4            2               2      2             5   
1             4            2               1      2             4   
2             0            2               1      2             1   
3             5            2               4      2             0   
4             5            2               4      2             4   
...         ...          ...             ...    ...           ...   
300148        5            1               4      0             2   
300149        5            1               0      0             5   
300150        5            1               1      0             5   
300151        5            1               1      0             2   
300152        5            1               4      0             2   

        destination_city  class  duration  
0                      5      1      2.17  
1                      5      1      2.33  
2                      5      1      2.17  
3                      5      1      2.25  
4                      5      1      2.33  
...                  ...    ...       ...  
300148                 3      0     10.08  
300149                 3      0     10.42  
300150                 3      0     13.83  
300151                 3      0     10.00  
300152                 3      0     10.08  

[300153 rows x 8 columns]
0         2
1         2
2         2
3         2
4         2
         ..
300148    1
300149    1
300150    1
300151    1
300152    1
Name: source_city, Length: 300153, dtype: int32
```

Out[16]:

	airline	source_city	departure_time	stops	arrival_time	destination_city	class	duration	price
0	4	2	2	2	5	5	1	2.17	5953
1	4	2	1	2	4	5	1	2.33	5953
2	0	2	1	2	1	5	1	2.17	5956
3	5	2	4	2	0	5	1	2.25	5955
4	5	2	4	2	4	5	1	2.33	5955
...	...	...	...	...	...	...	...	...	...
300148	5	1	4	0	2	3	0	10.08	69265
300149	5	1	0	0	5	3	0	10.42	77105
300150	5	1	1	0	5	3	0	13.83	79099
300151	5	1	1	0	2	3	0	10.00	81585
300152	5	1	4	0	2	3	0	10.08	81585

300153 rows × 9 columns

PCA Process

- PCA, is a dimensionality reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.
- Principal components analysis (PCA) is a standard tool in multivariate data analysis to reduce the number of dimensions, while retaining as much as possible of the data's variation.

What are the benefits of PCA analysis?

- Improve Algorithm Runtime.
- Improve Classification Accuracy.
- Visualization.
- Reduce Noise in Data.
- Feature Selection.

Applications of PCA:

- PCA is used to visualize multidimensional data.
- It is used to reduce the number of dimensions in healthcare data.
- PCA can help resize an image.
- It can be used in finance to analyze stock data and forecast returns.

Why is PCA preferable?

- The principal component method of factor analysis will help us.
- If you want to categorize the dependent & independent variables in your data, this algorithm will be our choice of consideration.
- Also, if you want to eliminate the noise components in your dimension analysis, PCA is the best computation method.

In [17]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
fit_pca = pca.fit_transform(X)
new_df = pd.DataFrame(data = fit_pca, columns = ['x', 'y'])
new_df
```

Out[17]:

	x	y
0	-10.122329	1.481031
1	-9.986488	1.682313
2	-10.151544	2.651835
3	-10.009926	2.732258
4	-9.921393	1.753761
...	...	...
300148	-2.091335	1.582142
300149	-1.832486	0.674615
300150	1.595032	0.720278
300151	-2.236634	1.451714
300152	-2.091335	1.582142

300153 rows × 2 columns

Splitting data for training and test

- Split the data set into two pieces — a training set and a testing set.
- This consists of random sampling without replacement about 80 percent of the rows (you can vary this) and putting them into your training set.
- The remaining 20 percent is put into your test set.

What is splitting data into training and testing?

- Data splitting is when data is divided into two or more subsets.
- Typically, with a two-part split, one part is used to evaluate or test the data and the other to train the model.

What is the best split for training and testing data?

- In general, keeping 80% of the data in the training set, 10% in the validation set & 10% in the test set is a good split to start with.

Training Data:

- The data you use to train an algorithm or machine learning model to predict the outcome you design your model to predict.
- If you are using supervised learning or some hybrid that includes that approach, your data will be enriched with data labeling or annotation.

Testing Data:

- Once your machine learning model is built (with your training data), you need unseen data to test your model.
- This data is called testing data, and you can use it to evaluate the performance and progress of your algorithms' training and adjust or optimize it for improved results.

Random State:

- It's a pseudo-random number parameter that allows you to reproduce the same train test split each time you run the code.

In [18]:
```
from sklearn.model_selection import train_test_split as trates
X_train, X_test, Y_train, Y_test = trates(X, Y, test_size=0.2, random_state=7)
```

Regression:

- A regression is a statistical technique that relates a dependent variable to one or more independent (explanatory) variables.
- A regression model is able to show whether changes observed in the dependent variable are associated with changes in one or more of the explanatory variables.
- It does this by essentially fitting a best-fit line and seeing how the data is dispersed around this line.
- Regression helps economists and financial analysts in things ranging from asset valuation to making predictions.
- In order for regression results to be properly interpreted, several assumptions about the data & the model itself must hold.

Linear regression:

- A data analysis technique that predicts the value of unknown data by using another related and known data value.
- It mathematically models the unknown or dependent variable and the known or independent variable as a linear equation.
- In statistics, linear regression is a statistical model which estimates the linear relationship between a scalar response and one or more explanatory variables (also known as dependent & independent variables).
- For Example, The weight of the person is linearly related to their height.

Why linear regression is used?

- Linear regression analysis is used to predict the value of a variable based on the value of another variable.
- The variable you want to predict is called the dependent variable.

Using Linear Regression

In [19]:
```
from sklearn.linear_model import LinearRegression

regressor=LinearRegression()
regressor.fit(X_train, Y_train)
```

Out[19]:
```
▾ LinearRegression

LinearRegression()
```

Prediction data test result

- Predictive analytics is the process of using data to forecast future outcomes.
- The process uses data analysis, ML, AI & statistical models to find patterns that might predict future behavior.

Accuracy:

- The accuracy function of the sklearn.metrics package package assigns subset accuracy in multi-label classification & calculates the accuracy score for a set of predicted labels against the true labels.
- Accuracy describes the model's behaviour across all classes.
- Informally, It's the fraction of predictions our model got right.
- **Formally, accuracy has the following definition: Accuracy = Number of correct predictions/Total number of predictions.**

In [20]:
```
y_pred = regressor.predict(X_test)
accuracy = regressor.score(X_test,Y_test)*100
print("ML model Accuracy is",accuracy,'%')
```

```
ML model Accuracy is 100.0 %
```

Mean Squared Error (MSE):

- The Mean Squared Error measures how close a regression line is to a set of data points.
- It is a risk function corresponding to the expected value of the squared error loss.
- **Mean square error is calculated by taking the average, specifically the mean, of errors squared from data as it relates to a function.**
- The value of the error ranges from zero to infinity.
- MSE increases exponentially with an increase in error.
- **A good model will have an MSE value closer to zero.**

Why mean square error is used?

- MSE is used to check how close estimates or forecasts are to actual values.
- **Lower the MSE, the closer is forecast to actual.**
- This is used as a model evaluation measure for regression models and the lower value indicates a better fit.

R2 Score:

- R-Squared (R²/The coefficient of determination) is a statistical measure in a regression model that determines the proportion of variance in the dependent variable that can be explained by the independent variable.
- Best possible score is 1.0 & it can be negative (because the model can be arbitrarily worse).
- R-squared is a goodness-of-fit measure for regression models.
- It is used to give us an idea about how well a regression model fits the observed data.

Evaluate Linear Regression

In [21]:
```
from sklearn.metrics import mean_squared_error,r2_score
from math import sqrt

print(Y_test)
print(y_pred)
print("RMSE = ", sqrt(mean_squared_error(Y_test,y_pred)))
print("r2_score: ",r2_score(Y_test,y_pred))
```

```
37442     2
26706     2
172802    3
240001    5
82508     5
         ..
210007    2
284192    3
104379    0
256618    0
84990     0
Name: source_city, Length: 60031, dtype: int32
[2.00000000e+00 2.00000000e+00 3.00000000e+00 ... 1.04804610e-13
 1.06204213e-13 9.92802217e-14]
RMSE =  6.978123543433147e-14
r2_score:  1.0
```

Support Vector Regression:

- SVM regression or Support Vector Regression (SVR) is a machine learning algorithm used for regression analysis.
- It is different from traditional linear regression methods as it finds a hyperplane that best fits the data points in a continuous space, instead of fitting a line to the data points.

- SVR has a great advantage in dealing with nonlinear processes by introducing a kernel function to project the original data into a high-dimensional linearly separable space.
- Since the separating hyperplane is supported (defined) by the vectors (data points) nearest the margin, so the algorithm is called SVM.
- The SVR algorithm aims to find the hyperplane that passes through as many data points as possible within a certain distance, called the margin.
- This approach helps to reduce the prediction error & allows SVR to handle non-linear relationships between input variables and the target variable using a kernel function.
- As a result, SVM regression is a powerful tool for regression tasks where there may be complex relationships between the input variables and the target variable.

Applications of SVM regression:

- SVM regression or Support Vector Regression (SVR) has a wide range of applications in various fields.
- It is commonly used in finance for predicting stock prices, in engineering for predicting machine performance & in bioinformatics for predicting protein structures.
- SVM regression is also used in natural language processing for text classification and sentiment analysis.
- Additionally, it is used in image processing for object recognition and in healthcare for predicting medical outcomes.
- Overall, SVM regression is a versatile algorithm that can be used in many domains for making accurate predictions.

Limitations of SVM:

- Long training time for large datasets. Difficult to understand and interpret the final model, variable weights & individual impact.
- Since the final model is not so easy to see, we can not do small calibrations to the model hence its tough to incorporate our business logic.
- SVM is comparatively less prone to overfitting.

Working Principle of SVM:

- SVM works by mapping data to a high-dimensional feature space so that data points can be categorized, even when the data are not otherwise linearly separable.
- A separator between the categories is found, then the data are transformed in such a way that the separator could be drawn as a hyperplane.

In [22]:
```
from sklearn import svm

regr = svm.SVR(kernel="linear")
regr.fit(X_train, Y_train)
```

Out[22]:
```
▾          SVR

SVR(kernel='linear')
```

Prediction data test result

In [23]:
```
y_pred = regr.predict(X_test)

print(Y_test)
print(y_pred)
```

```
37442     2
26706     2
172802    3
240001    5
82508     5
         ..
210007    2
284192    3
104379    0
256618    0
84990     0
Name: source_city, Length: 60031, dtype: int32
[2.02001017 2.020068   2.98015891 ... 0.09944401 0.09968661 0.10000427]
```

Evaluate SVR

In [24]:
```
print(Y_test)
print(y_pred)

print("RMSE SVR = ", sqrt(mean_squared_error(Y_test, y_pred)))
print("r2_score: ",r2_score(Y_test,y_pred))
```

```
37442     2
26706     2
172802    3
240001    5
82508     5
         ..
210007    2
284192    3
104379    0
256618    0
84990     0
Name: source_city, Length: 60031, dtype: int32
[2.02001017 2.020068   2.98015891 ... 0.09944401 0.09968661 0.10000427]
RMSE SVR =  0.06972792687222241
r2_score:  0.9984030719186624
```

KNN:

- KNN/K Nearest Neighbor is a Machine Learning algorithm that uses the similarity between our data to make classifications (supervised machine learning) or clustering (unsupervised machine learning).
- With KNN we can have a certain set of data and from it draw patterns that can classify or group our data.
- The concept of neighborhood depends on the idea that those close to us tend to be more like us.

KNN Regression

- K nearest neighbors is a simple algorithm that stores all available cases and predict the numerical target based on a similarity measure (e.g., distance functions).
- KNN has been used in statistical estimation & pattern recognition already in the beginning of 1970's as a non-parametric technique.

Advantages of KNN:

- Quick calculation time.
- Simple algorithm – to interpret.
- Versatile – useful for regression and classification.
- High accuracy – you do not need to compare with better-supervised learning models.
- No assumptions about data – no need to make additional assumptions, tune several parameters or build a model & This makes it crucial in nonlinear data case.

Disadvantages of KNN:

- Accuracy depends on the quality of the data.
- With large data, the prediction stage might be slow.
- Sensitive to the scale of the data and irrelevant features.
- Require high memory – need to store all of the training data.
- Given that it stores all of the training, it can be computationally expensive.

Summary of KNN Algorithm:

- K is a positive integer.
- With a new sample, you have to specify K.
- K is selected from database closest to the new sample.
- KNN doesn’t learn any model.
- KNN makes predictions using the similarity between an input sample and each training instance.

KNN is a great place to start when first learning to build models based on different data sets - Data set with a lot of different points & accurate information is your best place, to begin with KNN.

We should Keep these 3 points in mind:

- 1.A data set with lots of different points and labelled data is the ideal to use.
- 2.The best languages to use with KNN are R & python.
- 3.To find the most accurate results from your data set, you need to learn the correct practices for using this algorithm.

Using KNN Regression

In [25]:
```
from sklearn import neighbors
n_neighbors = 3

knn = neighbors.KNeighborsRegressor(n_neighbors, weights="uniform")
y_pred = knn.fit(X_train, Y_train).predict(X_test)
```

Evaluate KNN

In [26]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt

print(Y_test)
print(y_pred)

print("RMSE KNN Reg. = ",sqrt(mean_squared_error(Y_test, y_pred)))
print("r2_score: ",r2_score(Y_test,y_pred))
```

```
37442     2
26706     2
172802    3
240001    5
82508     5
         ..
210007    2
284192    3
104379    0
256618    0
84990     0
Name: source_city, Length: 60031, dtype: int32
[2. 2. 3. ... 0. 0. 0.]
RMSE KNN Reg. =  0.03146790622778793
r2_score:  0.9996747573085203
```

MLP Regression:

- A fully connected multi-layer neural network is called a Multilayer Perceptron (MLP).
- It has 3 layers including one hidden layer.
- If it has more than 1 hidden layer, it is called a deep ANN.
- An MLP is a typical example of a feedforward artificial neural network.

- MLPRegressor implements a multi-layer perceptron (MLP) that trains using backpropagation with no activation function in the output layer, which can also be seen as using the identity function as activation function.

What is MLP used for?

- The neurons in the MLP are trained with the back propagation learning algorithm.
- MLPs are designed to approximate any continuous function and can solve problems which are not linearly separable.
- The major use cases of MLP are pattern classification, recognition, prediction & approximation.

Advantages of MLP:

- It can be used to solve complex nonlinear problems.
- It handles large amounts of input data well.
- Makes quick predictions after training.
- The same accuracy ratio can be achieved even with smaller samples.

Limitations of MLP:

- Black box model: MLPs are often referred to as “black boxes” since it is sometimes unclear how the network makes its predictions or judgments.
- Overfitting: If the model is too complicated or the training data is too restricted, MLPs may easily overfit the training data.

What type of data is MLP best used for?

- We should use a Multilayer Perceptron (MLP) when working with structured data and tasks such as regression & classification. - Convolutional Neural Networks (CNNs) are well-suited for tasks involving image recognition & computer vision due to their ability to effectively capture spatial hierarchies.

Neural Network:

- A neural network is a method in artificial intelligence that teaches computers to process data in a way that is inspired by the human brain.
- It is a type of machine learning process, called deep learning, that uses interconnected nodes or neurons in a layered structure that resembles the human brain.
- A neural network contains layers of interconnected nodes.
- Each node is a known as perceptron & is similar to a multiple linear regression.
- The perceptron feeds the signal produced by a multiple linear regression into an activation function that may be nonlinear.

What is a Neuron in a Neural Networks?

- In the context of a neural network, a neuron is the most fundamental unit of processing.
- It's also called a perceptron.
- A neural network is based on the way a human brain works.
- So, we can say that it simulates the way the biological neurons signal to one another.

Benefits of Neural Networks:

- Neural networks offer a number of advantages, including requiring less formal statistical training, ability to implicitly detect complex nonlinear relationships between dependent & independent variables, ability to detect all possible interactions between predictor variables & the availability of multiple training algorithms.

MAX_ITER:

- The maximum number of major iterations for solution by the solver.

Random_state:

- Used to set the seed for the random generator so that we can ensure that the results that we get can be reproduced.
- Because of the nature of splitting the data in train & test is randomised you would get different data assigned to the train & test data unless you can control for the random factor.

Using MLP Regression

In [27]:
```
from sklearn.neural_network import MLPRegressor
regr = MLPRegressor(random_state=1, max_iter=5000).fit(X_train, Y_train)
y_pred = regr.predict(X_test)
```

Evaluate MLP

In [28]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt

print(Y_test)
print(y_pred)

print("RMSE MLP Reg. = ",sqrt(mean_squared_error(Y_test, y_pred)))
print("r2_score: ",r2_score(Y_test,y_pred))
```

```
37442     2
26706     2
172802    3
240001    5
82508     5
         ..
210007    2
284192    3
104379    0
256618    0
84990     0
Name: source_city, Length: 60031, dtype: int32
[ 2.00030731e+00  2.00464124e+00  3.00325530e+00 ... -1.86043309e-03
  4.61308968e-04  1.23559168e-03]
RMSE MLP Reg. =  0.003998498184381 92
r2_score:  0.9999947487141678
```

Conclusion

From the experimental results of several algorithms in regression, the RMSE values are produced as follows:

1. Linear Regression : **4.34**
2. SVR Regression : **0.069**
3. KNN Regression : **0.031**
4. MLP Regression : **0.015**

- **The smaller the RMSE value, the better.**
- Thus the **MLP Regression** algorithm is the best algorithm among the four algorithms tested.