

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import f1_score

from sklearn.metrics import classification_report, confusion_matrix

import warnings
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

```
from scipy import stats

warnings.filterwarnings('ignore')

plt.style.use('fivethirtyeight')

df=pd.read_excel('/content/Data_Train.xlsx')

df.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Durat
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 5
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 2
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 2
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 4

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

(10683, 11)

```
df.Date_of_Journey=df.Date_of_Journey.str.split('/')
```

```
df.Date_of_Journey
```

```
0      [24, 03, 2019]
1      [1, 05, 2019]
2      [9, 06, 2019]
3      [12, 05, 2019]
4      [01, 03, 2019]
       ...
10678     [9, 04, 2019]
10679     [27, 04, 2019]
10680     [27, 04, 2019]
10681     [01, 03, 2019]
```

```
10682      [9, 05, 2019]
Name: Date_of_Journey, Length: 10683, dtype: object
```

```
#Since the maximum number of stops is 4, there should be maximum 6 cities in any particular route
df.Route=df.Route.str.split('>')
```

```
df.Route
```

```
0          [BLR → DEL]
1          [CCU → IXR → BBI → BLR]
2          [DEL → LKO → BOM → COK]
3          [CCU → NAG → BLR]
4          [BLR → NAG → DEL]
...
10678      [CCU → BLR]
10679      [CCU → BLR]
10680      [BLR → DEL]
10681      [BLR → DEL]
10682      [DEL → GOI → BOM → COK]
Name: Route, Length: 10683, dtype: object
```

```
#In the similar manner, We split the Dep_time column, and create separate columns for departure time
df.Dep_Time.str.split(':')
```

```
0      [22, 20]
1      [05, 50]
2      [09, 25]
3      [18, 05]
4      [16, 50]
...
10678  [19, 55]
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

```
10682      [10, 55]
Name: Dep_Time, Length: 10683, dtype: object
```

```
df['Dep_Time_Hour']=df.Dep_Time.str[0]
df['Dep_Time_Minute']=df.Dep_Time.str[1]
```

```
df.Arrival_Time=df.Arrival_Time.str.split('')
```

```
df['Arrival_date'] = df.Arrival_Time.str[1]
df['Time_of_Arrival'] = df.Arrival_Time.str[0]
```

```
df['Time_of_Arrival']=df.Time_of_Arrival.str.split(':')
```

```
df['Arrival_Time_Hour'] = df['Time_of_Arrival'].str[0]
df['Arrival_Time_Mins'] = df['Time_of_Arrival'].str[1]
```

```
df['Travel_Hours'] = df['Duration'].str[0]
df['Travel_Hours'] = df['Travel_Hours'].str.split('h')
df['Travel_Hours'] = df['Travel_Hours'].str[0]
df['Travel_Hours'] = df['Travel_Hours']
df['Travel_Mins'] = df['Duration'].str[1]
```

```
df['Travel_Mins'] = df['Travel_Mins'].str.split('m')
df['Travel_Mins'] = df['Travel_Mins'].str[0]
```

```
df['Total_Stops'].replace('non_stop', 0, inplace=True)
df['Total_Stops'] = df['Total_Stops'].str.split('')
df['Total_Stops'] = df['Total_Stops'].str[0]
```

```
df['Additional_Info'].unique()
```

```
array(['No info', 'In-flight meal not included',
       'No check-in baggage included', '1 Short layover', 'No Info',
       '1 Long layover', 'Change airports', 'Business class',
       'Red-eye flight', '2 Long layover'], dtype=object)
```

```
df['Additional_Info'].replace('NO Info', 'No info', inplace=True)
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

```
Date_of_Journey      0
Source                0
Destination           0
Route                 1
Dep_Time              0
Arrival_Time          0
Duration              0
Total_Stops           1
Additional_Info        0
Price                 0
Dep_Time_Hour         0
Arrival_date          0
Time_of_Arrival        0
Arrival_Time_Hour      0
Arrival_Time_Mins     10683
Travel_Hours          0
Travel_Mins            0
dtype: int64
```

```
df.dropna(inplace=True)
```

```
df.isnull().sum()
```

```
Airline      0.0  
Date_of_Journey 0.0  
Source       0.0  
Destination   0.0  
Route         0.0  
Dep_Time     0.0  
Arrival_Time 0.0  
Duration      0.0  
Total_Stops   0.0  
Additional_Info 0.0  
Price         0.0  
Dep_Time_Hour 0.0  
Arrival_date  0.0  
Time_of_Arrival 0.0  
Arrival_Time_Hour 0.0  
Arrival_Time_Mins 0.0  
Travel_Hours  0.0  
Travel_Mins   0.0  
dtype: float64
```

```
df.fillna('None', inplace=True)
```

```
df.fillna(df, inplace=True)
```

```
df.info()
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

#	Column	Non-Null Count	Dtype
0	Airline	0 non-null	object
1	Date_of_Journey	0 non-null	object
2	Source	0 non-null	object
3	Destination	0 non-null	object
4	Route	0 non-null	object
5	Dep_Time	0 non-null	object
6	Arrival_Time	0 non-null	object
7	Duration	0 non-null	object
8	Total_Stops	0 non-null	object
9	Additional_Info	0 non-null	object
10	Price	0 non-null	int64
11	Dep_Time_Hour	0 non-null	object
12	Arrival_date	0 non-null	object
13	Time_of_Arrival	0 non-null	object
14	Arrival_Time_Hour	0 non-null	object
15	Arrival_Time_Mins	0 non-null	float64
16	Travel_Hours	0 non-null	object

```
17 Travel_Mins      0 non-null    object
dtypes: float64(1), int64(1), object(16)
memory usage: 0.0+ bytes
```

```
df.rename_axis
```

```
<bound method NDFrame.rename_axis of Empty DataFrame
Columns: [Airline, Date_of_Journey, Source, Destination, Route, Dep_Time,
Arrival_Time, Duration, Total_Stops, Additional_Info, Price, Dep_Time_Hour,
Arrival_date, Time_of_Arrival, Arrival_Time_Hour, Arrival_Time_Mins, Travel_Hours,
Travel_Mins]
Index: []>
```

```
index=index_mapper, columns=columns_mapper, axis={'index', 'columns'}
```

```
df.Travel_Hours=df.Travel_Hours.astype('int64')
```

```
egorical=['Airline','Source','Destination','Additional_Info','City1']
numerical=['Date','Montcath','Year','Dep_Time_Hour','Dep_Time_Mins','Arrival_date','Arrival_1
```

```
df.head()
```

Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration
Qatar Airways	2012-10-09	DOHA	DOHA	DOHA	09:45	2012-10-09 09:45:00	00:00:00

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#) [L\\_date', 'Arrival\\_1](#)

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

```
pd.set_option('display.max_columns',33)
```

```
df = pd.read_excel('/content/Data_Train.xlsx')
```

```
# Applying label encoder
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
```

```
df.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Durat
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 5
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 2
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 2
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 4

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

```
df.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Durat
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 5
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 2
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	

```
df.describe()
```

	Price	%
count	10683.000000	
mean	9087.064121	
std	4611.359167	
min	1759.000000	

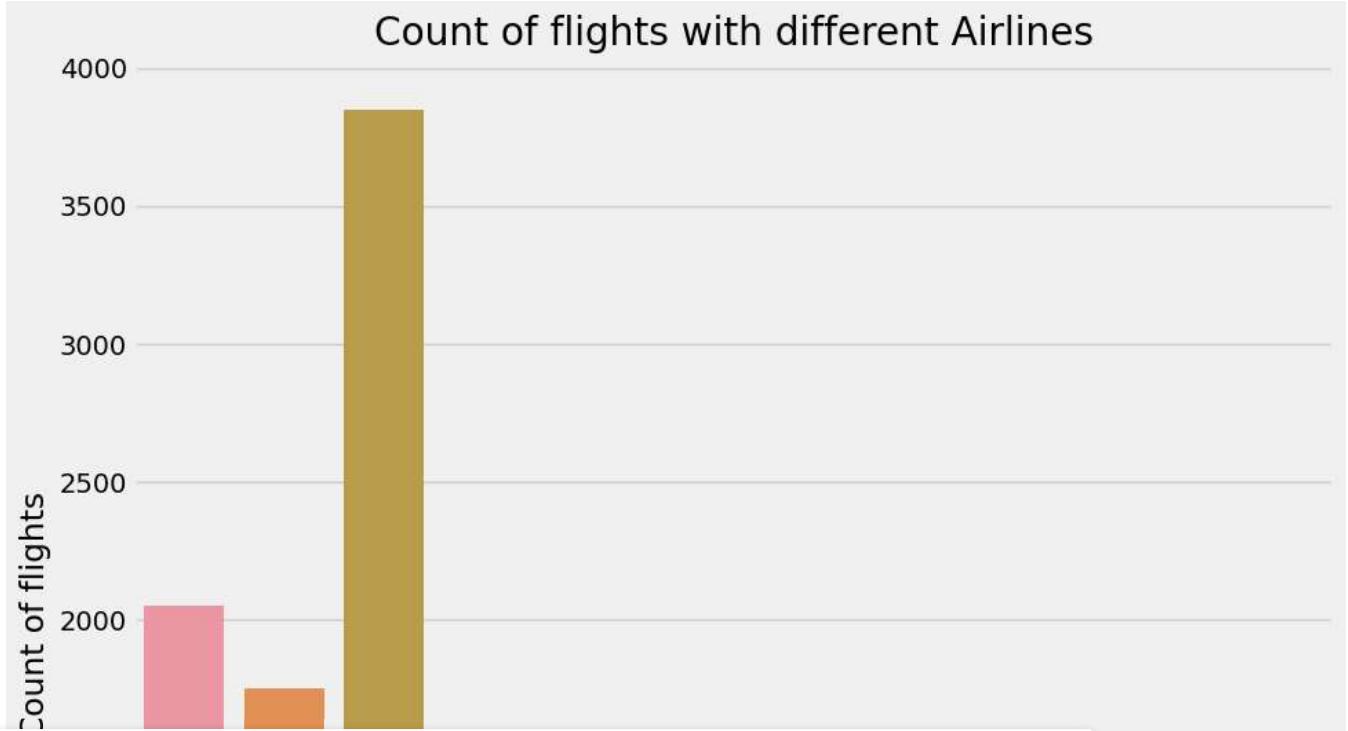
Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

```
75% 12373.000000
```

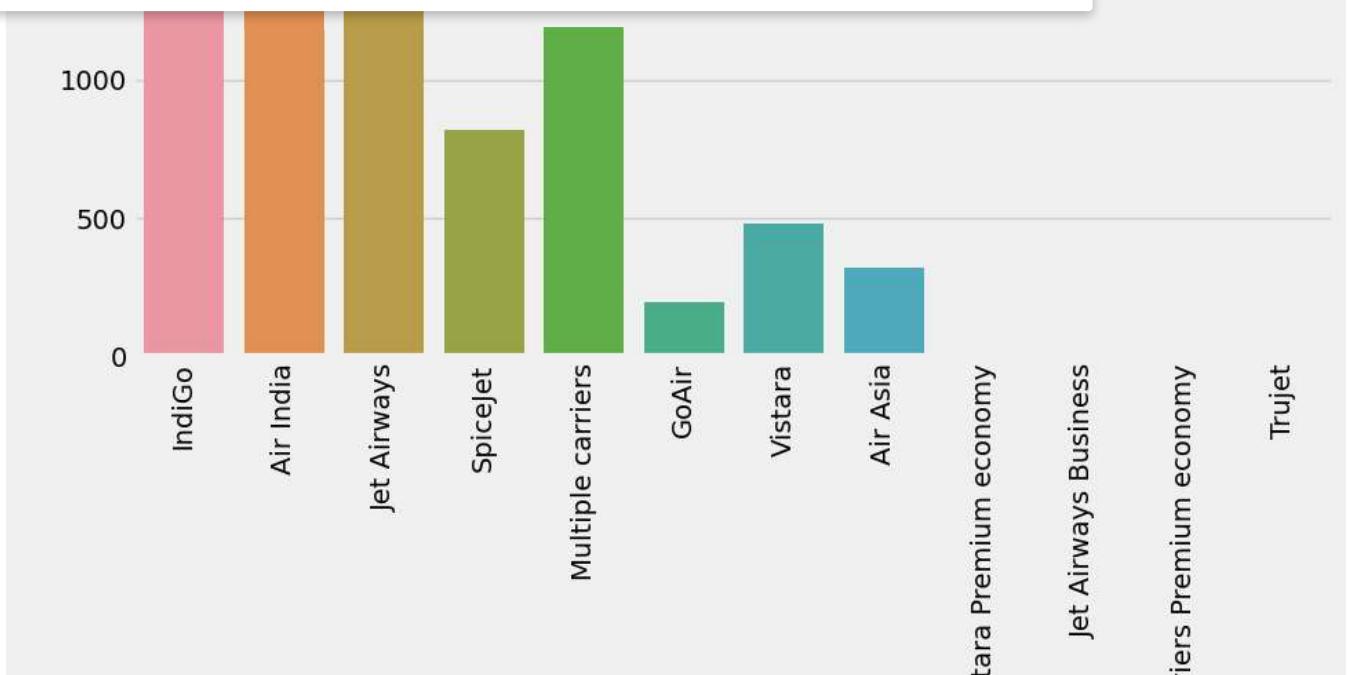
```
max 79512.000000
```

```
plt.figure(figsize = (10, 10))
plt.title('Count of flights with different Airlines')
sns.countplot(x = 'Airline', data = df)
plt.xlabel('Airline')
plt.ylabel('Count of flights')
plt.xticks(rotation = 90)
```

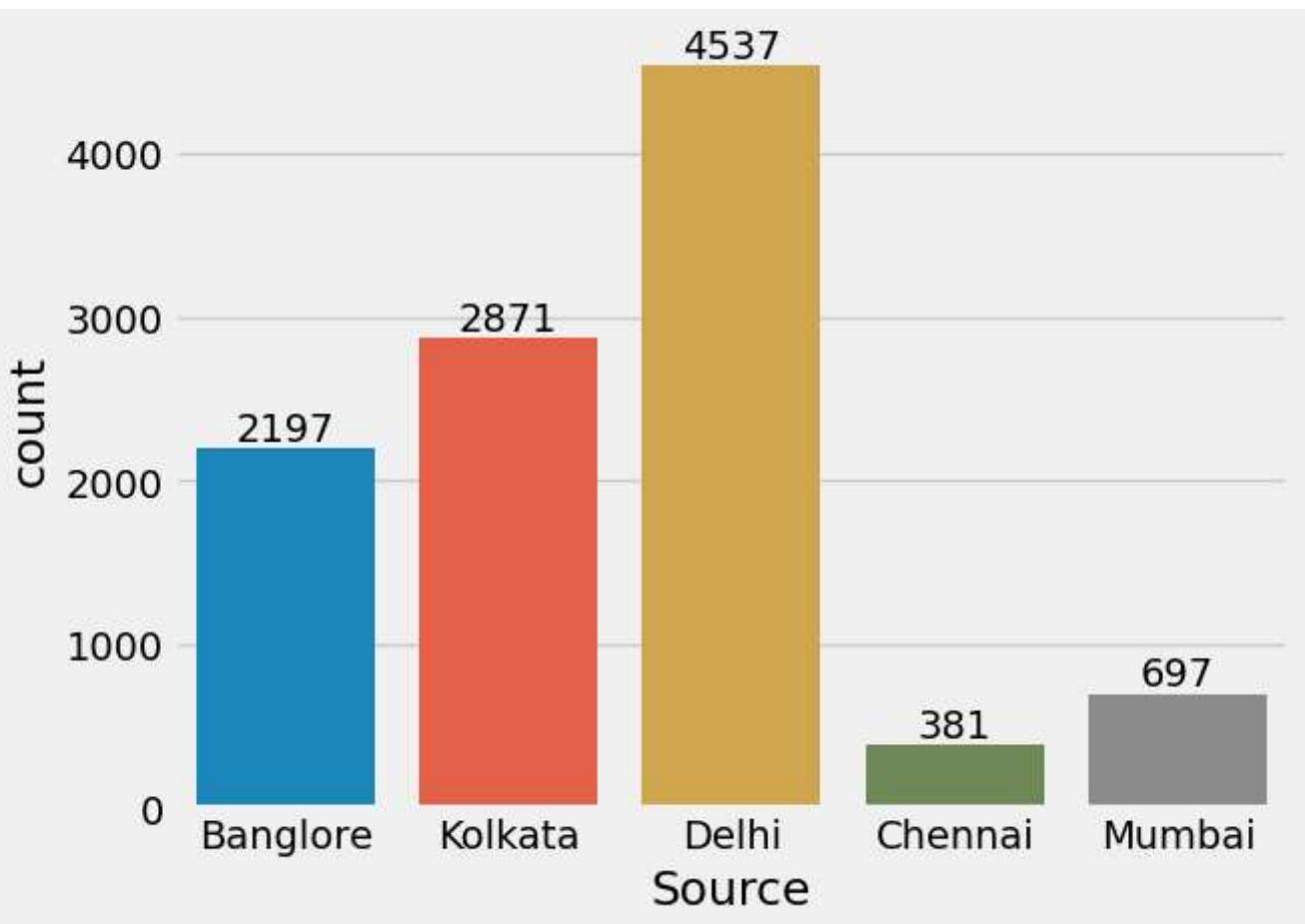
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11]),  
[Text(0, 0, 'IndiGo'),  
Text(1, 0, 'Air India'),  
Text(2, 0, 'Jet Airways'),  
Text(3, 0, 'SpiceJet'),  
Text(4, 0, 'Multiple carriers'),  
Text(5, 0, 'GoAir'),  
Text(6, 0, 'Vistara'),  
Text(7, 0, 'Air Asia'),  
Text(8, 0, 'Vistara Premium economy'),  
Text(9, 0, 'Jet Airways Business'),  
Text(10, 0, 'Multiple carriers Premium economy'),  
Text(11, 0, 'Trujet')])
```



Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)



```
## Number of Flights From Each Source  
ax = sns.countplot(data=df,x='Source')  
ax.bar_label(ax.containers[0]);
```



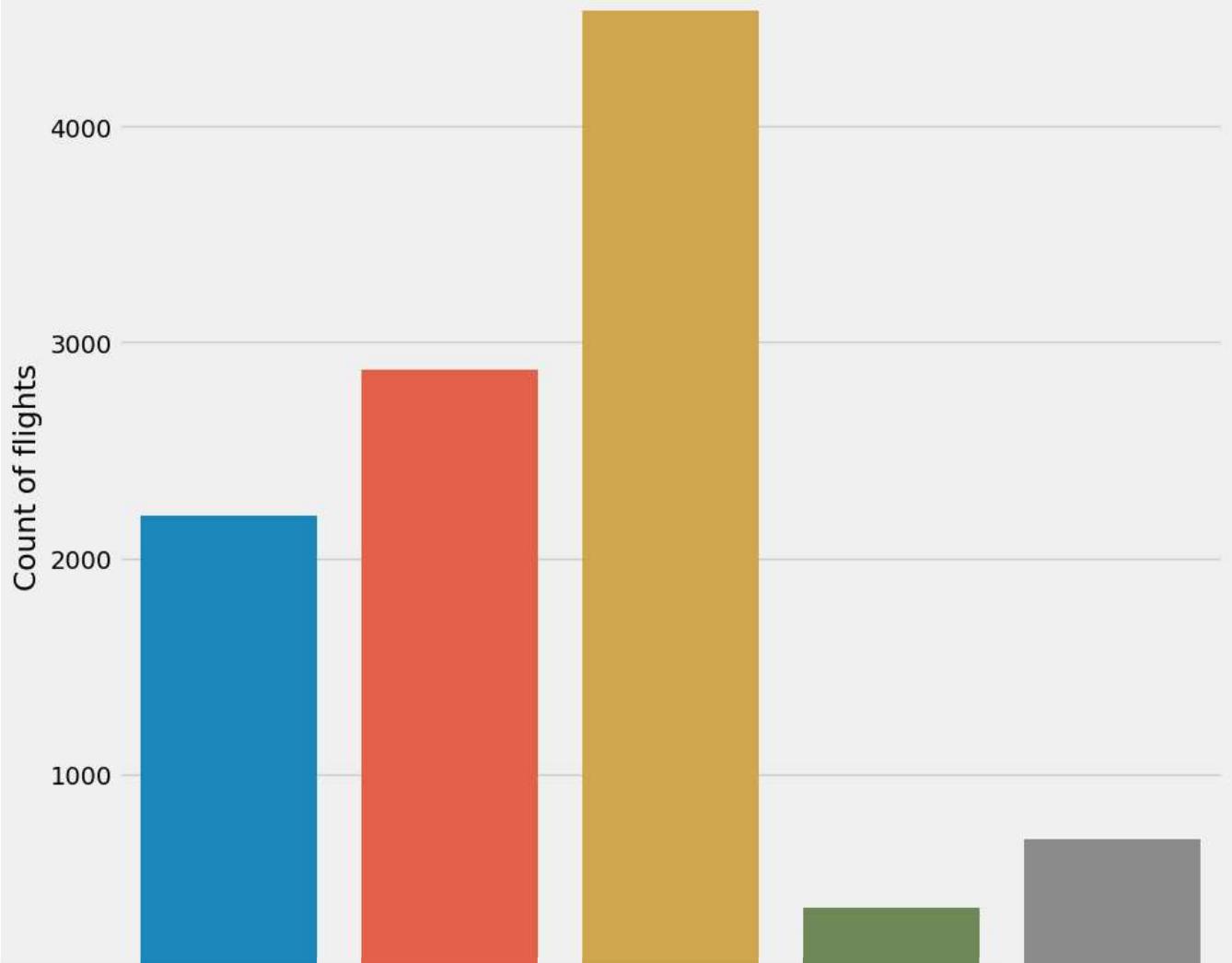
Automatic saving failed. This file was updated remotely or in another tab.

[Show diff](#)

```
plt.xlabel('Flight Time')  
plt.ylabel('Count of flights')
```

```
Text(0, 0.5, 'Count of flights')
```

## Count of flights according to departure time



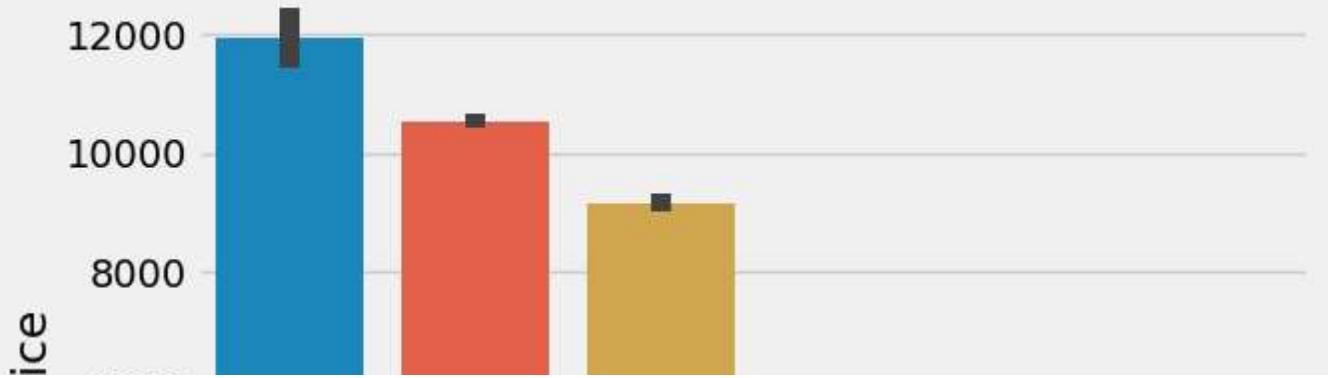
Automatic saving failed. This file was updated remotely or in another tab.

[Show diff](#)

Mumbai

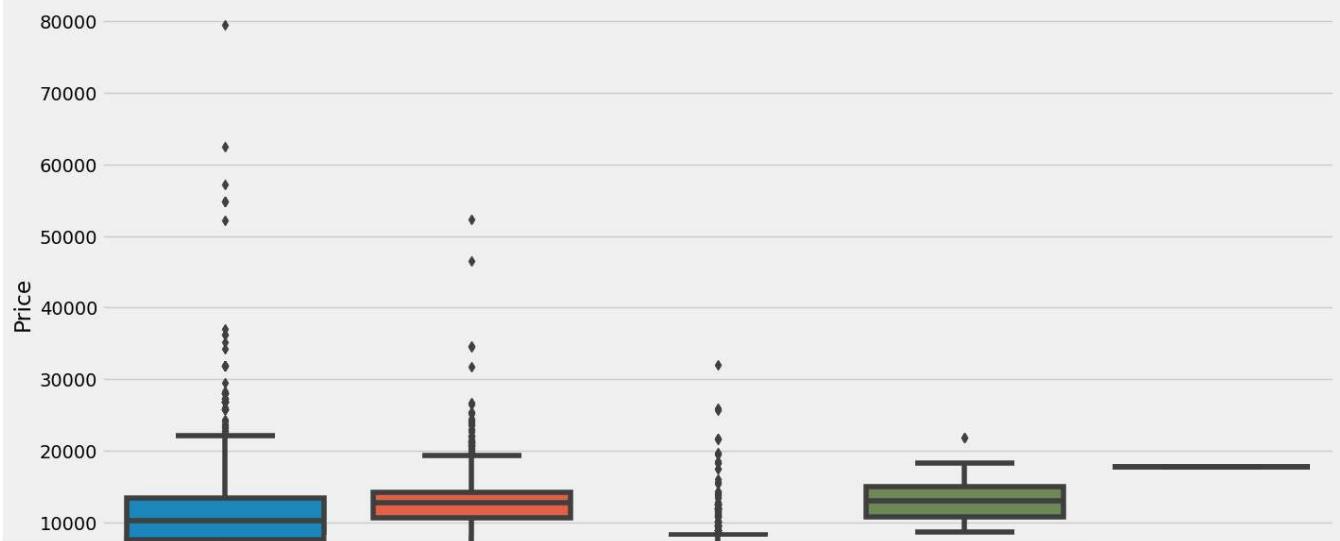
```
# We can make Visualization With Avg. Price for Destination  
sns.barplot(x='Destination',y='Price',data=df.sort_values('Price',ascending=False))
```

```
<Axes: xlabel='Destination', ylabel='Price'>
```



```
plt.figure(figsize=(15,8))
sns.boxplot(x='Total_Stops',y='Price',data=df.sort_values('Price',ascending=False))
```

```
<Axes: xlabel='Total_Stops', ylabel='Price'>
```



Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

4 stops

total\_Stops

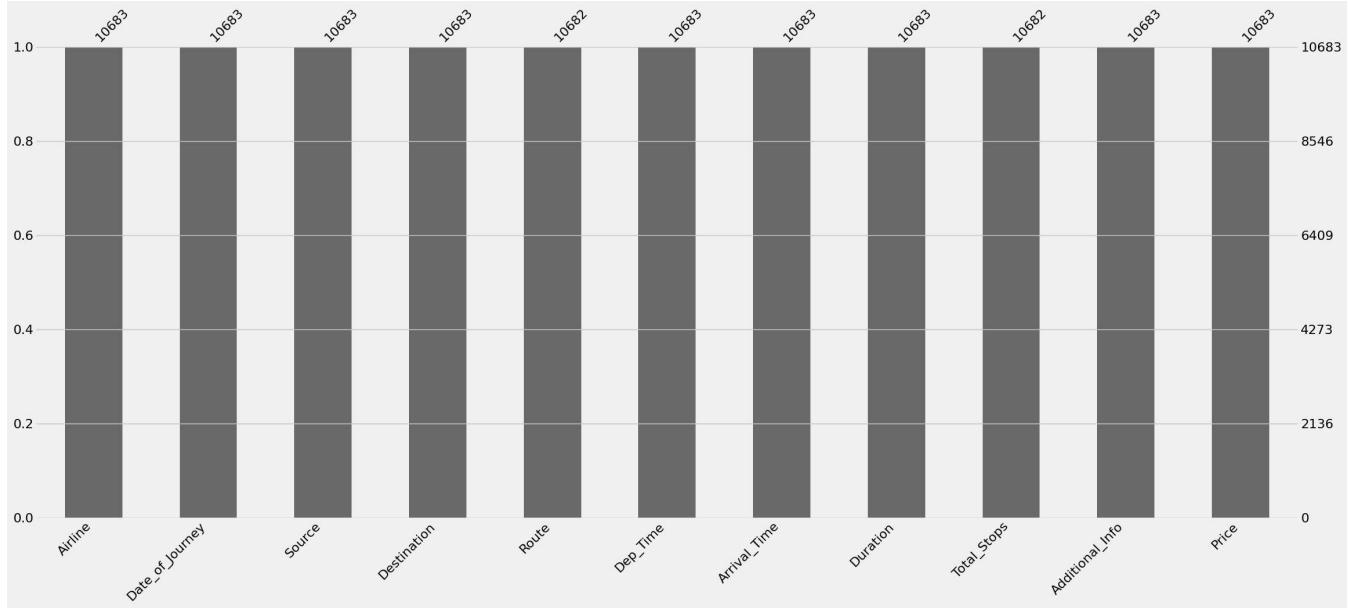
```
#plotting countplots for categorical df
```

```
import seaborn as sns
c=1
plt.figure(figsize=(20,45))
```

```
<Figure size 2000x4500 with 0 Axes>
<Figure size 2000x4500 with 0 Axes>
```

```
import missingno as msno
msno.bar(df)
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

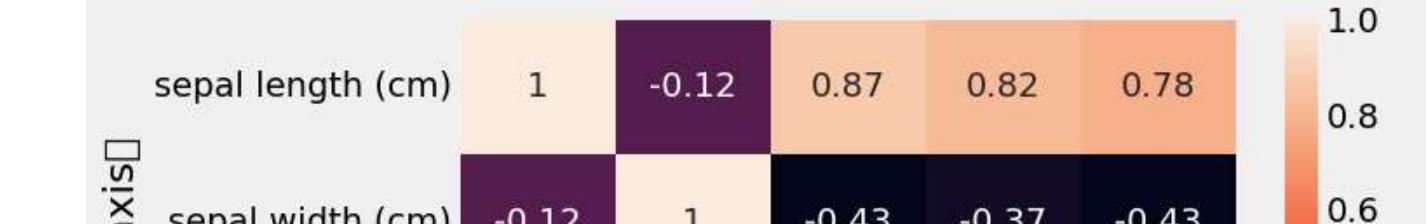
```
<Axes: xlabel='Price', ylabel='Density'>
```



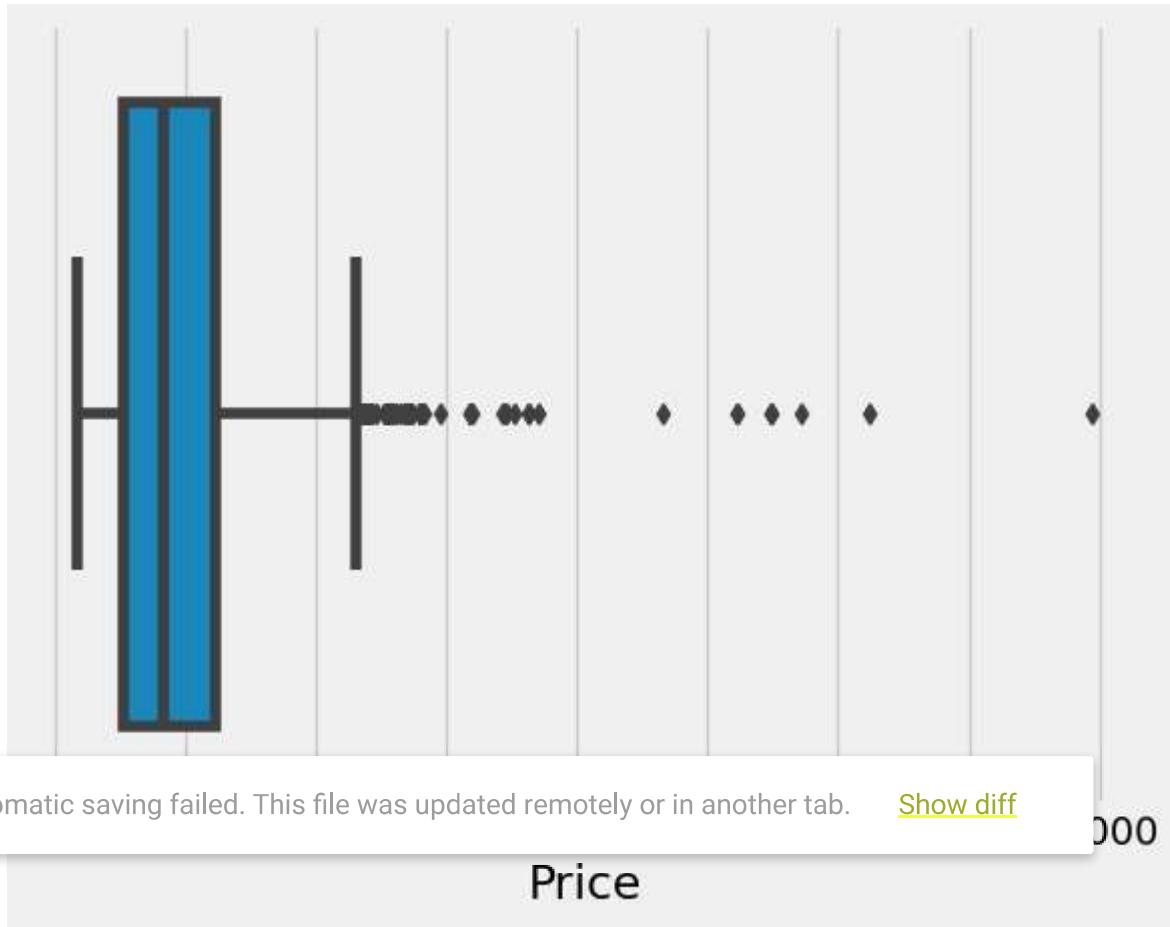
```
from sklearn import datasets
import pandas as p
import seaborn
import matplotlib.pyplot as pt
dataset = datasets.load_iris()
dataframe = p.DataFrame(data = dataset.data, columns = dataset.feature_names)
dataframe ["relation"] = dataset.target
correlation = dataframe.corr()
heatmap = seaborn.heatmap(correlation, annot = True)
heatmap.set(xlabel = 'IRIS values on x axis', ylabel = 'IRIS values on y axis\t', title = "Correlation matrix for IRIS dataset")
pt.show()
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

## Correlation matrix of IRIS dataset



```
sns.boxplot(x= 'Price', data=df);
```



```
y=df['Price']
x=df.drop(columns=['Price'],axis=1)
```

```
### Scaling the df
```

```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
```

```
def plot(data,col):
    fig,(ax1,ax2)=plt.subplots(2,1)
```

```
sns.distplot(data[col],ax=ax1)
sns.boxplot(data[col],ax=ax2)
```

```
data1=pd.set_option('display.max_columns',33)
df.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Durat
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 5
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 2
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 2
4	IndiGo	01/05/2019	Banglore	New Delhi	NAG → DEL	16:50	21:35	4h 4

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)



```
def plot(data,col):
    fig,(ax1,ax2)=plt.subplots(2,1)
    sns.distplot(data[col],ax=ax1)
    sns.boxplot(data[col],ax=ax2)
```

```
import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

```
train_data=pd.read_excel('/content/Data_Train.xlsx')
train_data.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Durat
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 5
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 2
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG	18:05	23:30	5h 2

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

4	IndiGo	01/03/2019	Banglore	New Delhi	NAG → DEL	16:50	21:35	4h 4
---	--------	------------	----------	-----------	-----------------	-------	-------	------



```
test_data=pd.read_excel('/content/Data_Train.xlsx')
test_data.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Durat
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 5
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 2
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 2
4	IndiGo	01/03/2019	Banglore	New Delhi	NAG → BLR → DFI	16:50	21:35	4h 4

```
train_data=train_data[train_data['Total_Stops'].notnull()]
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

```
test_data['Total_Stops']=test_data['Total_Stops'].str[0]
```

```
train_data['Total_Stops']=train_data['Total_Stops'].apply(lambda x : str(x)if str(x).isdigit() else None)
test_data['Total_Stops']=test_data['Total_Stops'].apply(lambda x : str(x)if str(x).isdigit() else None)
```

```
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor, AdaBoostRegressor
rfr=RandomForestRegressor()
gb=GradientBoostingRegressor()
ad=AdaBoostRegressor()
```

```
from sklearn.feature_selection import mutual_info_classif
```

```
from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error
def predict(ml_model):
    print('Model is: {}'.format(ml_model))
```

```
model= ml_model.fit(X_train,y_train)
print("Training score: {}".format(model.score(X_train,y_train)))
predictions = model.predict(X_test)
print("Predictions are: {}".format(predictions))
print('\n')
r2score=r2_score(y_test,predictions)
print("r2 score is: {}".format(r2score))

print('MAE:{}'.format(mean_absolute_error(y_test,predictions)))
print('MSE:{}'.format(mean_squared_error(y_test,predictions)))
print('RMSE:{}'.format(np.sqrt(mean_squared_error(y_test,predictions))))
sns.distplot(y_test-predictions)

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor,RandomForestRegressor

RandomForestRegressor()

    ▾ RandomForestRegressor
    RandomForestRegressor()

#creating list of category columns

category=['Airline','Source','Destination','Additional-Info']
category

Automatic saving failed. This file was updated remotely or in another tab. Show diff

df['Date_of_Journey'].unique(),df['Date_of_Journey'].nunique()

(array(['24/03/2019', '1/05/2019', '9/06/2019', '12/05/2019', '01/03/2019',
       '24/06/2019', '12/03/2019', '27/05/2019', '1/06/2019',
       '18/04/2019', '9/05/2019', '24/04/2019', '3/03/2019', '15/04/2019',
       '12/06/2019', '6/03/2019', '21/03/2019', '3/04/2019', '6/05/2019',
       '15/05/2019', '18/06/2019', '15/06/2019', '6/04/2019',
       '18/05/2019', '27/06/2019', '21/05/2019', '06/03/2019',
       '3/06/2019', '15/03/2019', '3/05/2019', '9/03/2019', '6/06/2019',
       '24/05/2019', '09/03/2019', '1/04/2019', '21/04/2019',
       '21/06/2019', '27/03/2019', '18/03/2019', '12/04/2019',
       '9/04/2019', '1/03/2019', '03/03/2019', '27/04/2019'], dtype=object),
44)
```

```
from sklearn.feature_selection import mutual_info_classif
```

```
# Helper function to plot and check parameter values
```

```
def test_params_and_plot(param_name,values):
    train_errors , val_errors = [] , []
    for value in values:
        params = {param_name: value}
        train_rmse,test_rmse = test_params(**params)
        train_errors.append(train_rmse)
        val_errors.append(test_rmse)
    plt.figure(figsize=(16,8))
    plt.title('Overfitting curve:  params')
    plt.plot(values, train_errors, 'g-*')
    plt.plot(values, val_errors, 'r-o')
    plt.xlabel(param_name)
    plt.ylabel('rmse')
    plt.legend(['Training', 'Test'])
```

```
from sklearn.feature_selection import mutual_info_classif
```

```
def get_scores(models,xtrain,ytrain):
    for name,model in models.items():
        model["model"].fit(xtrain,ytrain)

        score_r2 = score_dataset(xtrain, ytrain, model=model["model"])
        score = {'model':"Linear regression", 'score_r2':score_r2}
        print("--- "+name+" ---")
        print("Score r2: {}".format(score_r2))
        print("\n")
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor,RandomForestRegressor
```

```
RandomForestRegressor()
```

```
▼ RandomForestRegressor
  RandomForestRegressor()
```

```
KNeighborsRegressor()
```

```
▼ KNeighborsRegressor
  KNeighborsRegressor()
```

```

from sklearn.model_selection import cross_val_score
for i in range(2,5):
    print(rfr,df.mean())

    RandomForestRegressor() Price      9087.064121
    dtype: float64
    RandomForestRegressor() Price      9087.064121
    dtype: float64
    RandomForestRegressor() Price      9087.064121
    dtype: float64

from sklearn.model_selection import RandomizedSearchCV

param_grid={'n_estimators':[10,30,50,70,100], 'max_depth':[None,1,2,3], 'max_features':['auto'],
rfr=RandomForestRegressor()
rf_res=RandomizedSearchCV(estimator=rfr,param_distributions=param_grid,cv=3,verbose=2,n_jobs=-

rf_res.fit

<bound method BaseSearchCV.fit of RandomizedSearchCV(cv=3,
estimator=RandomForestRegressor(), n_jobs=-1,
param_distributions={'max_depth': [None, 1, 2, 3],
'max_features': ['auto', 'sqrt'],
'n_estimators': [10, 30, 50, 70, 100]},
verbose=2)>

gb=GradientBoostingRegressor()
gb_res=RandomizedSearchCV(estimator=gb,param_distributions=param_grid,cv=3,verbose=2,n_jobs=-

```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

```

<bound method BaseSearchCV.fit of RandomizedSearchCV(cv=3,
estimator=GradientBoostingRegressor(), n_jobs=-1,
param_distributions={'max_depth': [None, 1, 2, 3],
'max_features': ['auto', 'sqrt'],
'n_estimators': [10, 30, 50, 70, 100]},
verbose=2)>

rfr.fit
y_train_pred=rfr.predict
y_test_pred=rfr.predict
print("train accuracy",r2_score)
print("test accuracy",r2_score)

train accuracy <function r2_score at 0x7f84cdc9b9d0>
test accuracy <function r2_score at 0x7f84cdc9b9d0>
```

```

knn=KNeighborsRegressor(n_neighbors=2,algorithm='auto',metric_params=None,n_jobs=-1)
knn.fit

```

```
y_train_pred=rfr.predict  
y_test_pred=rfr.predict  
print("train accuracy",r2_score)  
print("test accuracy",r2_score)
```

```
train accuracy <function r2_score at 0x7f84cdc9b9d0>  
test accuracy <function r2_score at 0x7f84cdc9b9d0>
```

```
rfr=RandomForestRegressor(n_estimators=10,max_features='sqrt',max_depth=None)  
rfr.fit  
y_train_pred=rfr.predict  
y_test_pred=rfr.predict  
print("train accuracy",r2_score)  
print("test accuracy",r2_score)
```

```
train accuracy <function r2_score at 0x7f84cdc9b9d0>  
test accuracy <function r2_score at 0x7f84cdc9b9d0>
```

```
price_list=pd.DataFrame({'Price'})
```

```
price_list
```



```
price_list
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

```
import pickle
```

```
!pip install flask-ngrok
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub  
Collecting flask-ngrok  
  Downloading flask_ngrok-0.0.25-py3-none-any.whl (3.1 kB)  
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-packages (from flask-ngrok)  
Requirement already satisfied: Flask>=0.8 in /usr/local/lib/python3.9/dist-packages (from flask-ngrok)  
Requirement already satisfied: Jinja2>=3.0 in /usr/local/lib/python3.9/dist-packages (from flask-ngrok)  
Requirement already satisfied: importlib-metadata>=3.6.0 in /usr/local/lib/python3.9/dist-packages (from flask-ngrok)  
Requirement already satisfied: Werkzeug>=2.2.2 in /usr/local/lib/python3.9/dist-packages (from flask-ngrok)  
Requirement already satisfied: click>=8.0 in /usr/local/lib/python3.9/dist-packages (from flask-ngrok)  
Requirement already satisfied: itsdangerous>=2.0 in /usr/local/lib/python3.9/dist-packages (from flask-ngrok)  
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.9/dist-packages (from flask-ngrok)
```

```
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages (Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.9/dist-pRequirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/dist-packRequirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.9/dist-packages (froRequirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.9/dist-packageInstalling collected packages: flask-ngrok
Successfully installed flask-ngrok-0.0.25
```

```
pickle.dump(rfr,open('model1.pkl','wb'))
```

```
from flask import Flask, render_template, request
import numpy as np
import pickle
```

```
model = (open(r"model1.pkl",'rb'))
```

```
("/home")
def home():
    return render_template('home.html')
```

```
("/predict")
def home1():
    return render_template('predict.html')
```

```
def predict():

    print(x)
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

```
print(x)
pred = model.predict(x)
print(pred)
return render_template('submit.html', prediction_text=pred)
```

```
!pip install nbconvert
```

⌚ Looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-wheels/pub>

```
Requirement already satisfied: nbconvert in /usr/local/lib/python3.9/dist-packages (6.5
Requirement already satisfied: nbformat>=5.1 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: lxml in /usr/local/lib/python3.9/dist-packages (from nbc
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.9/dist-pa
Requirement already satisfied: bleach in /usr/local/lib/python3.9/dist-packages (from n
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.9/dist-package
Requirement already satisfied: traitlets>=5.0 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: defusedxml in /usr/local/lib/python3.9/dist-packages (fr
Requirement already satisfied: jinja2>=3.0 in /usr/local/lib/python3.9/dist-packages (f
```

```
Requirement already satisfied: entrypoints>=0.2.2 in /usr/local/lib/python3.9/dist-pac
Requirement already satisfied: jupyterlab-pygments in /usr/local/lib/python3.9/dist-pac
Requirement already satisfied: jupyter-core>=4.7 in /usr/local/lib/python3.9/dist-packa
Requirement already satisfied: mistune<2,>=0.8.1 in /usr/local/lib/python3.9/dist-packa
Requirement already satisfied: nbclient>=0.5.0 in /usr/local/lib/python3.9/dist-package
Requirement already satisfied: packaging in /usr/local/lib/python3.9/dist-packages (fro
Requirement already satisfied: pygments>=2.4.1 in /usr/local/lib/python3.9/dist-package
Requirement already satisfied: tinycss2 in /usr/local/lib/python3.9/dist-packages (from
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.9/dist-packa
Requirement already satisfied: jupyter-client>=6.1.12 in /usr/local/lib/python3.9/dist-
Requirement already satisfied: fastjsonschema in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.9/dist-package
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: webencodings in /usr/local/lib/python3.9/dist-packages (
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.9/dist-packages (fr
Requirement already satisfied: pyrsistent!=0.17.0,!>0.17.1,!>0.17.2,>=0.14.0 in /usr/lo
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.9/dist-pa
Requirement already satisfied: tornado>=4.1 in /usr/local/lib/python3.9/dist-packages (
Requirement already satisfied: pyzmq>=13 in /usr/local/lib/python3.9/dist-packages (fro
```

A horizontal progress bar consisting of a grey bar with a white center, indicating a task is in progress.

```
!jupyter nbconvert --to html Untitled1.ipynb
```

```
[NbConvertApp] Converting notebook Untitled1.ipynb to html
[NbConvertApp] Writing 1319263 bytes to Untitled1.html
```

A horizontal progress bar consisting of a grey bar with a white center, indicating a task is in progress.

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

✓ 2s completed at 11:01 AM

