SCM-516
Team Final Project-Team 407

Team Members
1. Anwesha Banerjee
2. Qinxin Zhang
3. Alvin Lee
4. Satyanand Eranki
5. Xuzhou Feng

Classification

Dataset

Bank Marketing

Donated on 2/13/2012

The data is related with direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict if the client will subscribe a term deposit (variable y).

Dataset Characteristics	Subject Area	Associated Tasks
Multivariate	Business	Classification
Feature Type	**# Instances**	**# Features**
Categorical, Integer	45211	16

- Moro, S., Rita, P., & Cortez, P. (2014). Bank Marketing [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C5K306.

Data Preprocessing

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
2424	29	blue-collar	single	secondary	no	783	yes	no	unknown	13	may	166	1	-1	0	unknown	no
34231	33	admin.	married	tertiary	no	69	no	yes	cellular	4	may	77	2	-1	0	unknown	no
26140	40	entrepreneur	married	secondary	no	1349	no	no	cellular	20	nov	82	3	-1	0	unknown	no
25200	36	services	single	secondary	no	461	no	no	cellular	18	nov	434	2	-1	0	unknown	no
9167	48	technician	married	unknown	no	460	no	no	unknown	5	jun	99	3	-1	0	unknown	no

Output after Data Processing-Part 1

	age	job	marital	education	default	balance	housing	loan	duration	campaign	y
23787	47	blue-collar	married	primary	no	961	no	no	45	11	no
36427	37	technician	single	tertiary	no	555	yes	no	180	3	no
1547	53	blue-collar	married	secondary	no	359	no	yes	238	8	no
33652	29	blue-collar	married	secondary	no	457	yes	no	72	3	no
27706	30	blue-collar	single	tertiary	no	4127	yes	no	47	4	no

Data Preprocessing contd…

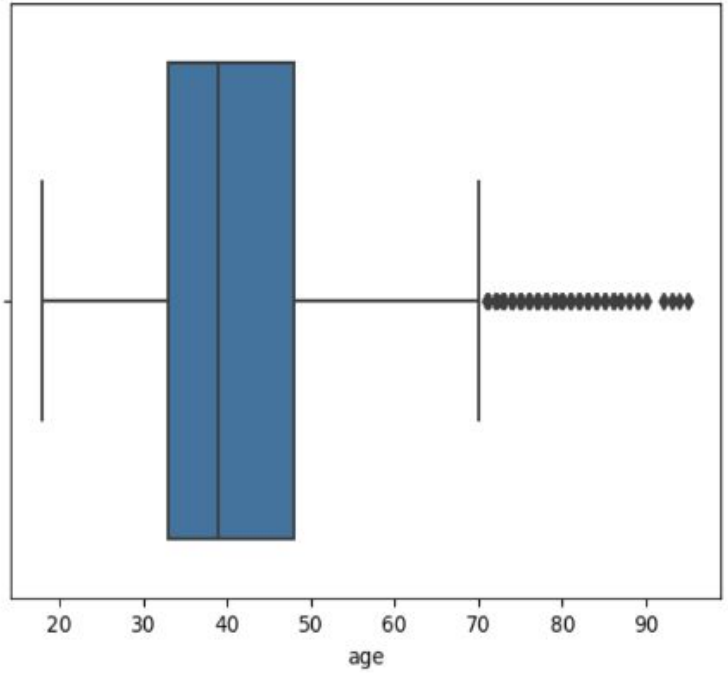
Checking for “Missing Values”

```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
duration     0
campaign     0
y            0
dtype: int64
```

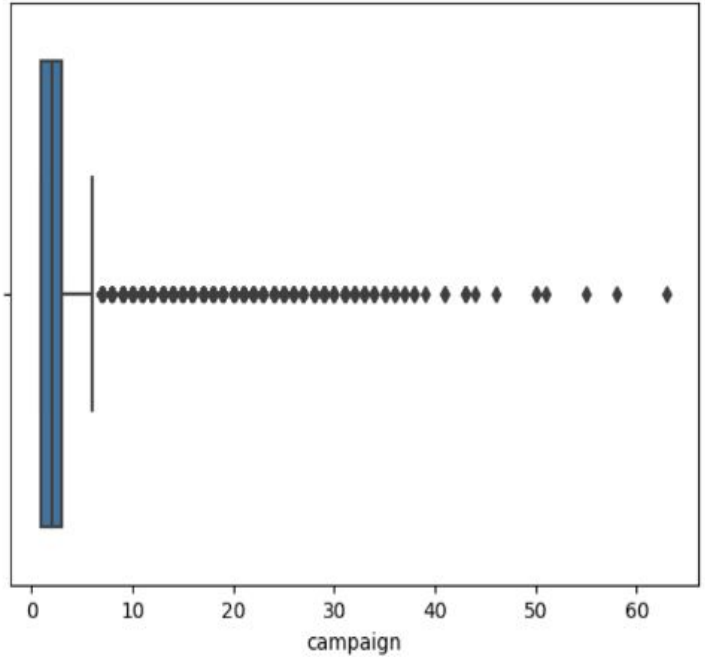
In our data set it is seen that we do not have any “missing values”

Data Preprocessing contd…

Removing “Outliers”



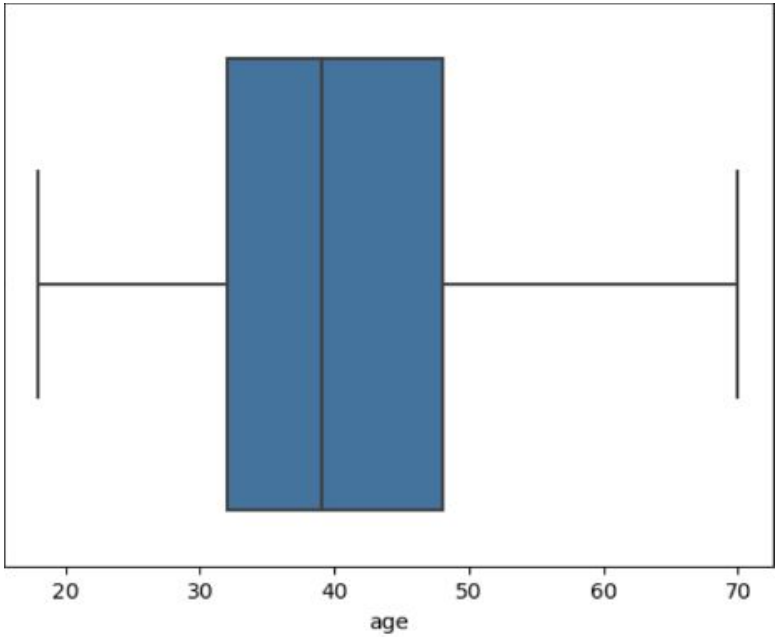
Removing outliers for “Age”



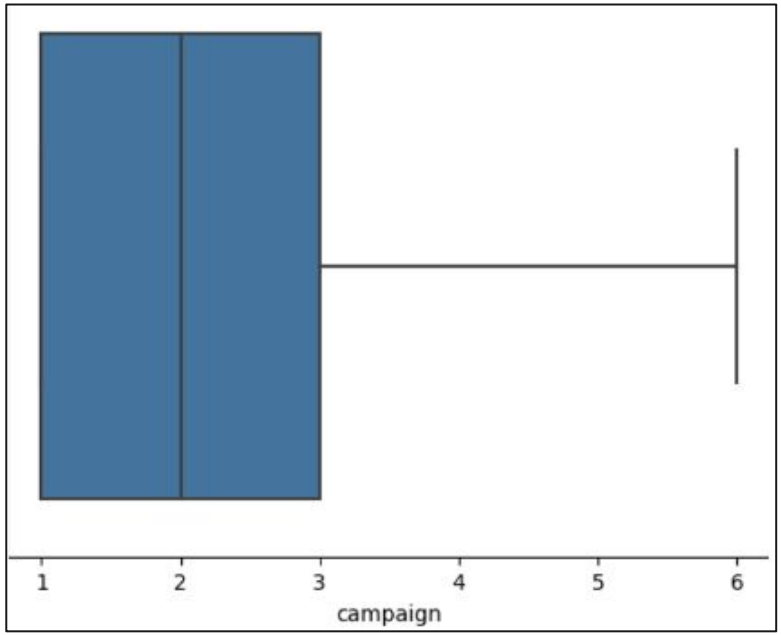
Removing outliers for “Campaign”

Data Preprocessing contd…

After Removing “Outliers”



Result of Removing outliers for “Age”



Result of Removing outliers for “Campaign”

Encoding & Standardization

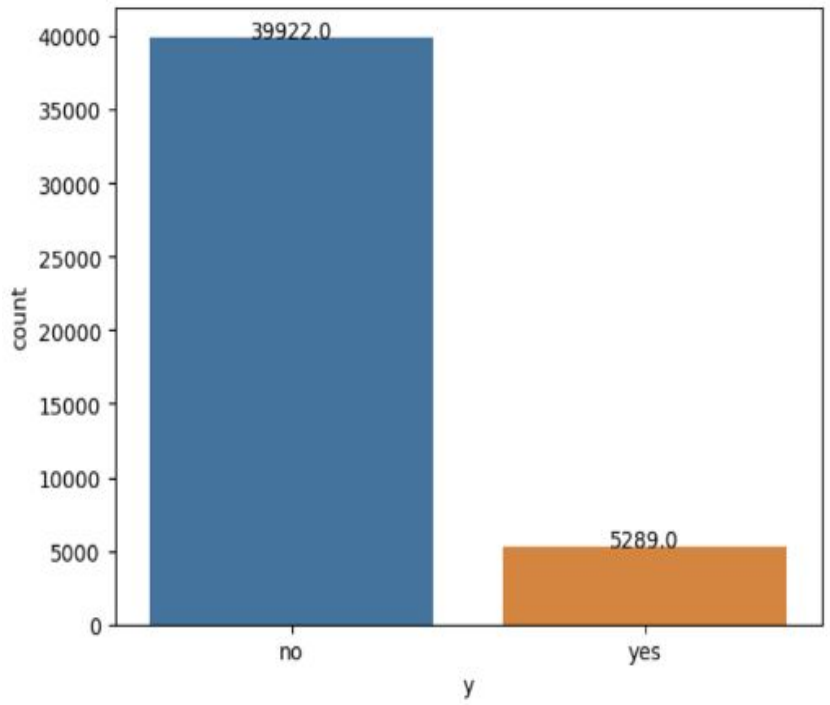
Data set on which we are encoding and standardizing

	age	job	marital	education	default	balance	housing	loan	duration	campaign	y
23787	47	blue-collar	married	primary	no	961	no	no	45	11	no
36427	37	technician	single	tertiary	no	555	yes	no	180	3	no
1547	53	blue-collar	married	secondary	no	359	no	yes	238	8	no
33652	29	blue-collar	married	secondary	no	457	yes	no	72	3	no
27706	30	blue-collar	single	tertiary	no	4127	yes	no	47	4	no

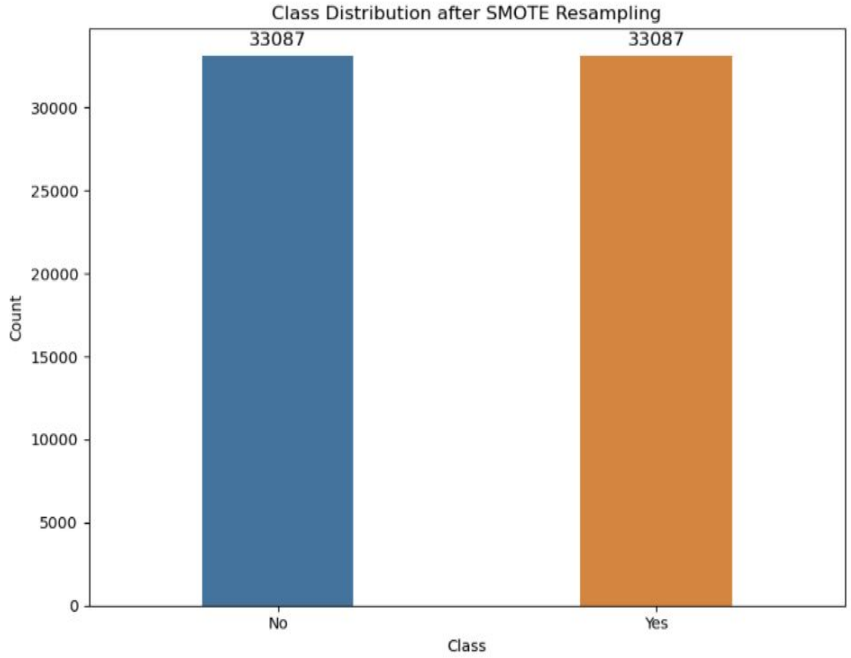
Output

	age	job	marital	education	default	balance	housing	loan	duration	campaign	y
41748	0.269231	4	2	2	0	0.916055	1	0	0.069054	0.0	0
24558	0.653846	4	1	2	0	0.602861	1	0	0.029116	0.0	0
15520	0.269231	9	2	1	0	0.795219	1	0	0.062613	0.0	0
40919	0.538462	9	1	1	0	0.354602	0	0	0.027570	0.0	0
35534	0.365385	1	1	1	0	0.493883	1	0	0.111827	0.4	0

SMOTE Sampling Technique



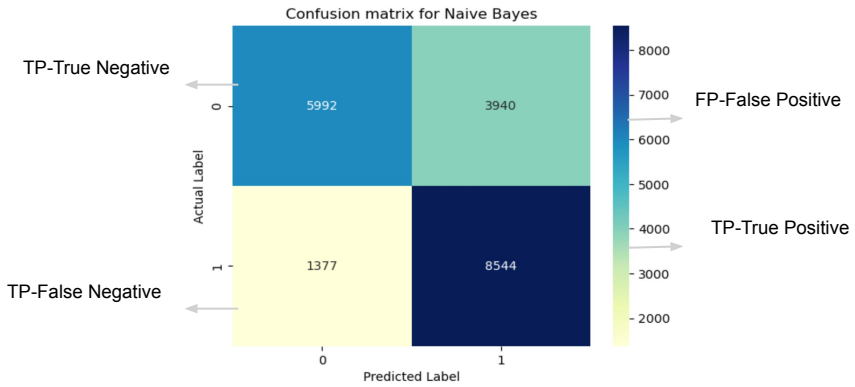
Pre-Sampling

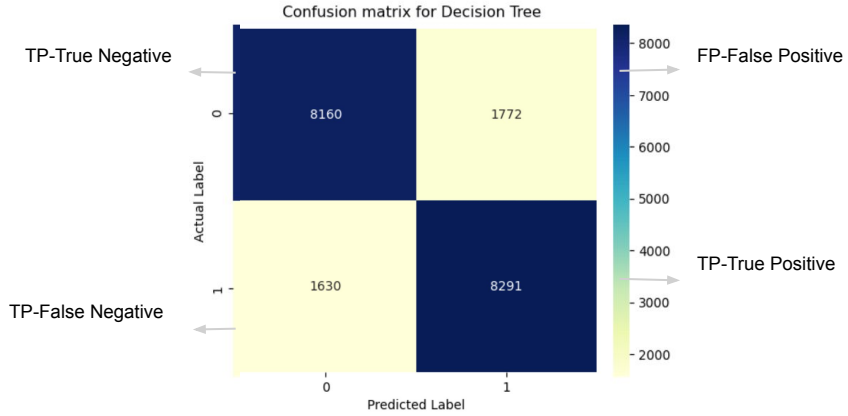


Post-Sampling

Confusion Matrix(Accuracy, Precision, Recall, and F1 Score)+Optimal K-Value

Confusion Matrix





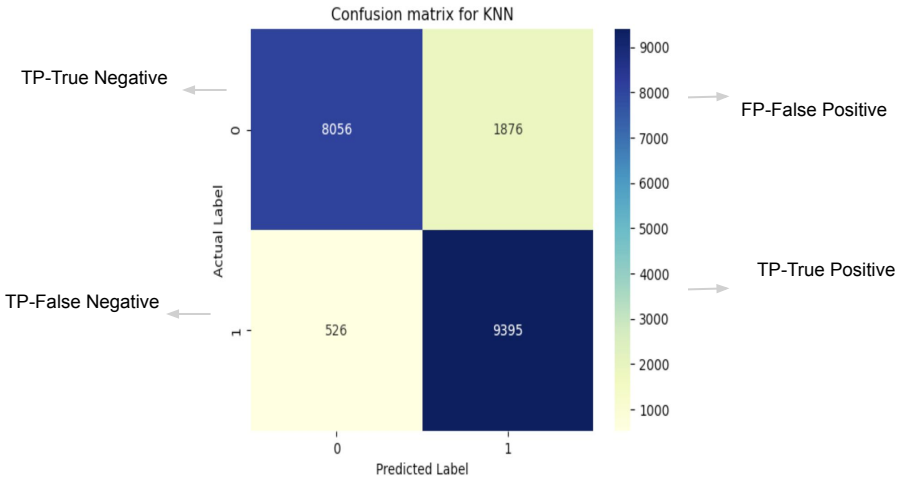
Model: Naive Bayes

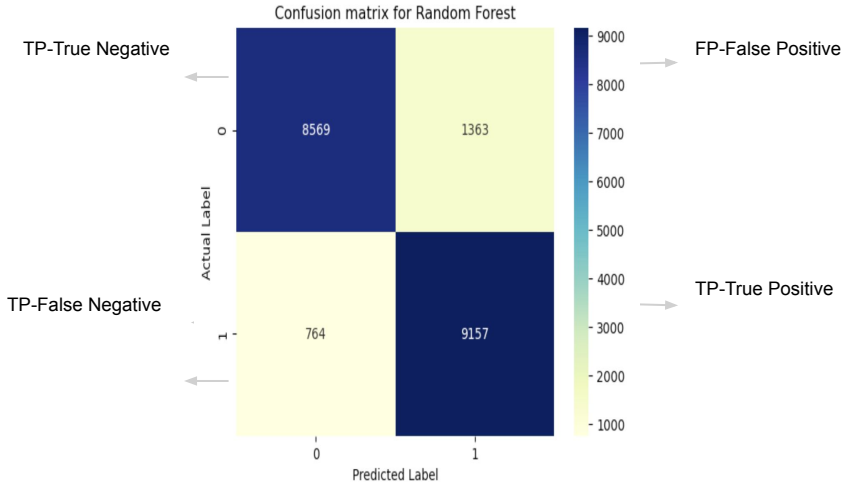
Confusion Matrix:[[5992 3940],[1377 8544]]
Accuracy: 0.73
Precision: 0.68
PRecall: 0.86
F1 Score: 0.76

Model: Decision Tree

Confusion Matrix:[[8160 1772], [1630 8291]]
Accuracy: 0.83
Precision: 0.82
PRecall: 0.84
F1 Score: 0.83

Confusion Matrix





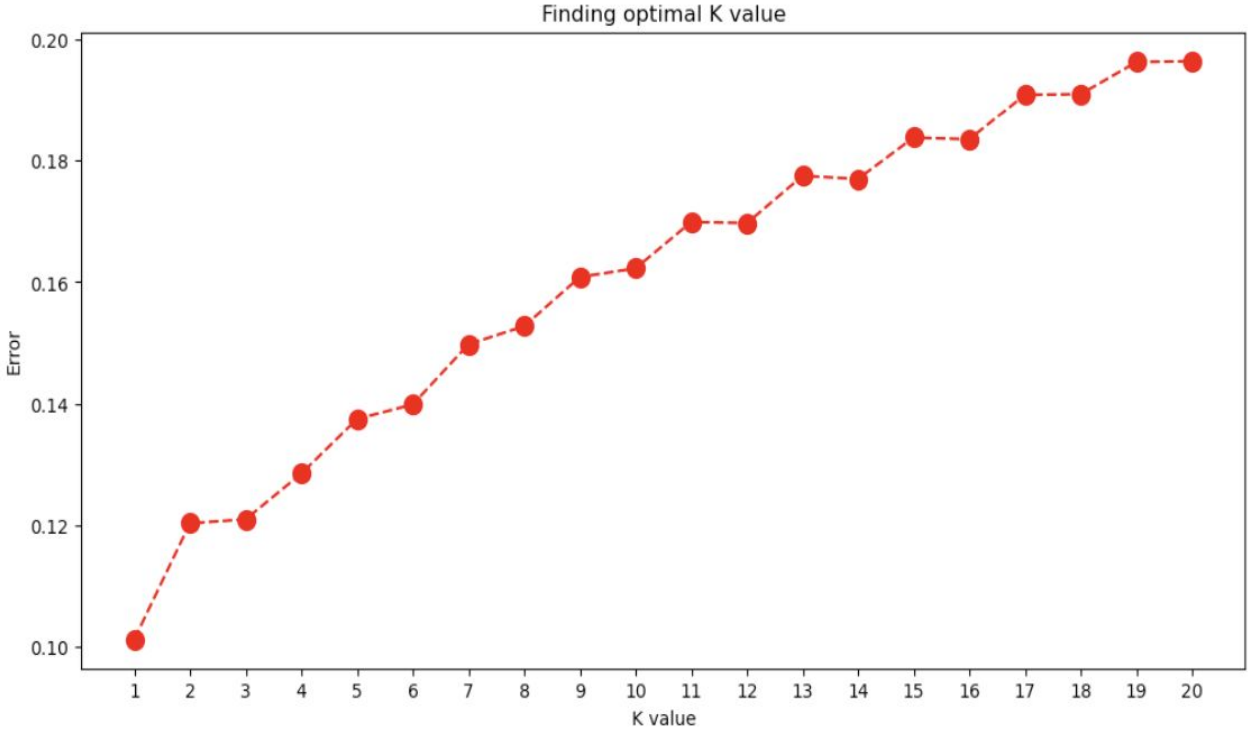
Model: KNN

Confusion Matrix:[[8056 1876],[526 9395]]
Accuracy: 0.88
Precision: 0.83
PRecall: 0.95
F1 Score: 0.89

✅ **Model: Random Forest**

Confusion Matrix:[[8569 1363], [764 9157]]
Accuracy: 0.89
Precision: 0.87
PRecall: 0.92
F1 Score: 0.90

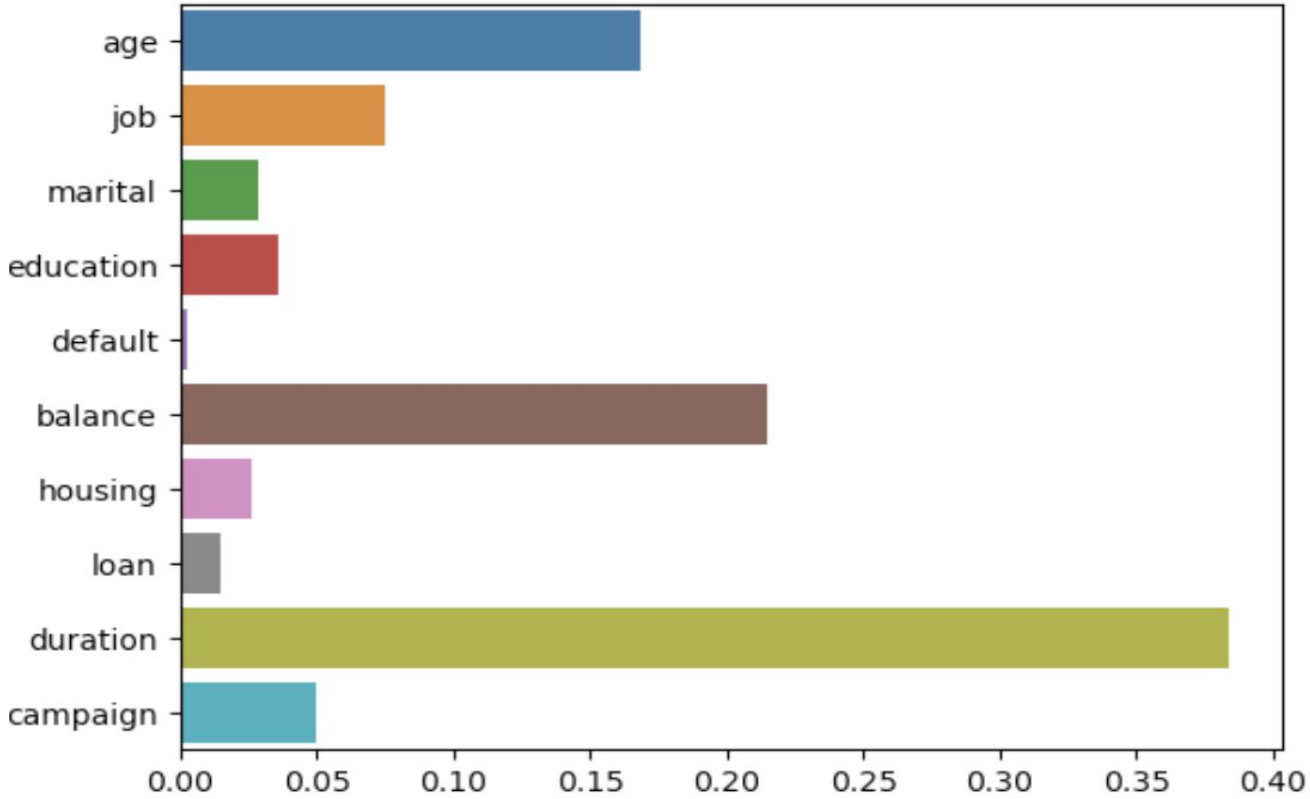
To find the Optimal K-Value



Optimal K-Value-3

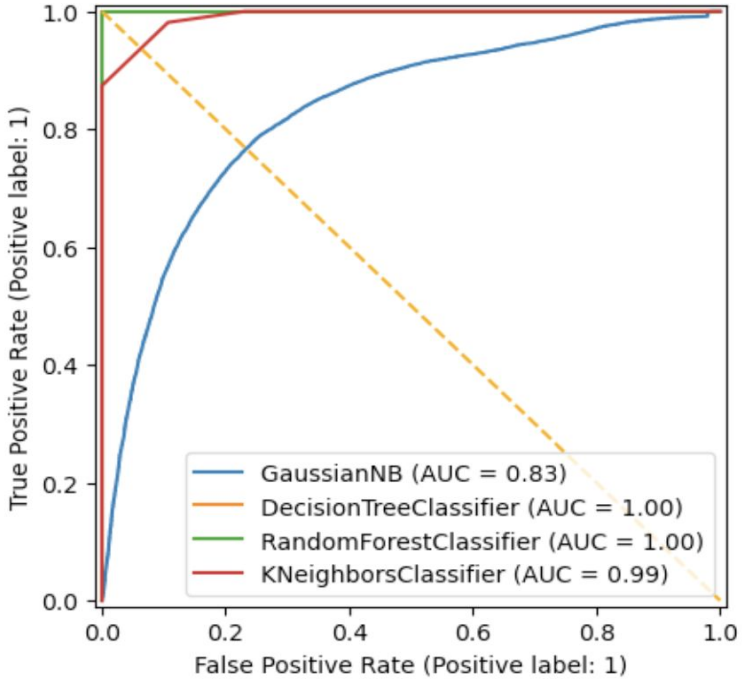
Feature Importance of Random Forest

Feature Importance for Random Forest

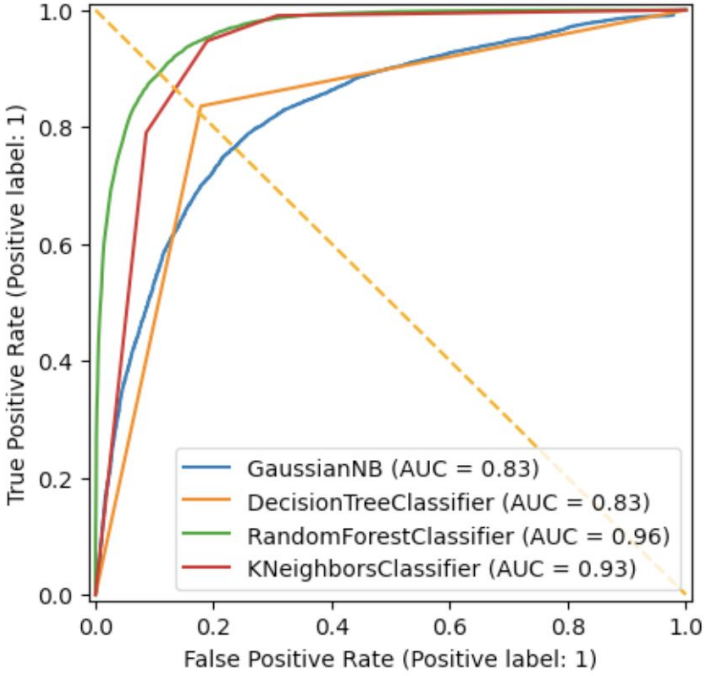


ROC

ROC



ROC-Training Dataset



ROC-Testing Dataset

Sample Data - Testing

Sample Data Testing

Sample Data Set

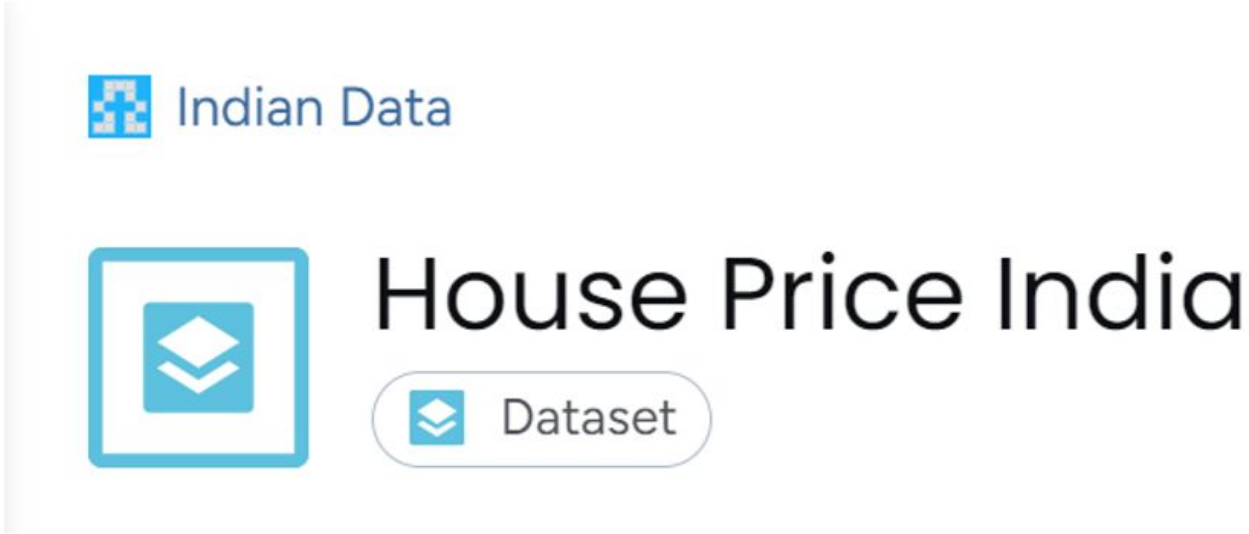
	age	job	marital	education	default	balance	housing	loan	duration	campaign
0	35	teacher	single	tertiary	no	1200	yes	no	320	3
1	28	artist	divorced	secondary	no	850	no	yes	145	2
2	52	entrepreneur	married	tertiary	yes	2100	yes	yes	180	1
3	43	engineer	married	secondary	no	3500	no	no	400	4

Testing Data Set - O/P

	age	job	marital	education	default	balance	housing	loan	duration	campaign	pred_NB	pred_DT	pred_KNN	pred_RF
0	35	teacher	single	tertiary	no	1200	yes	no	320	3	yes	no	no	yes
1	28	artist	divorced	secondary	no	850	no	yes	145	2	no	no	no	no
2	52	entrepreneur	married	tertiary	yes	2100	yes	yes	180	1	no	yes	no	yes
3	43	engineer	married	secondary	no	3500	no	no	400	4	yes	yes	no	yes

Regression

Dataset

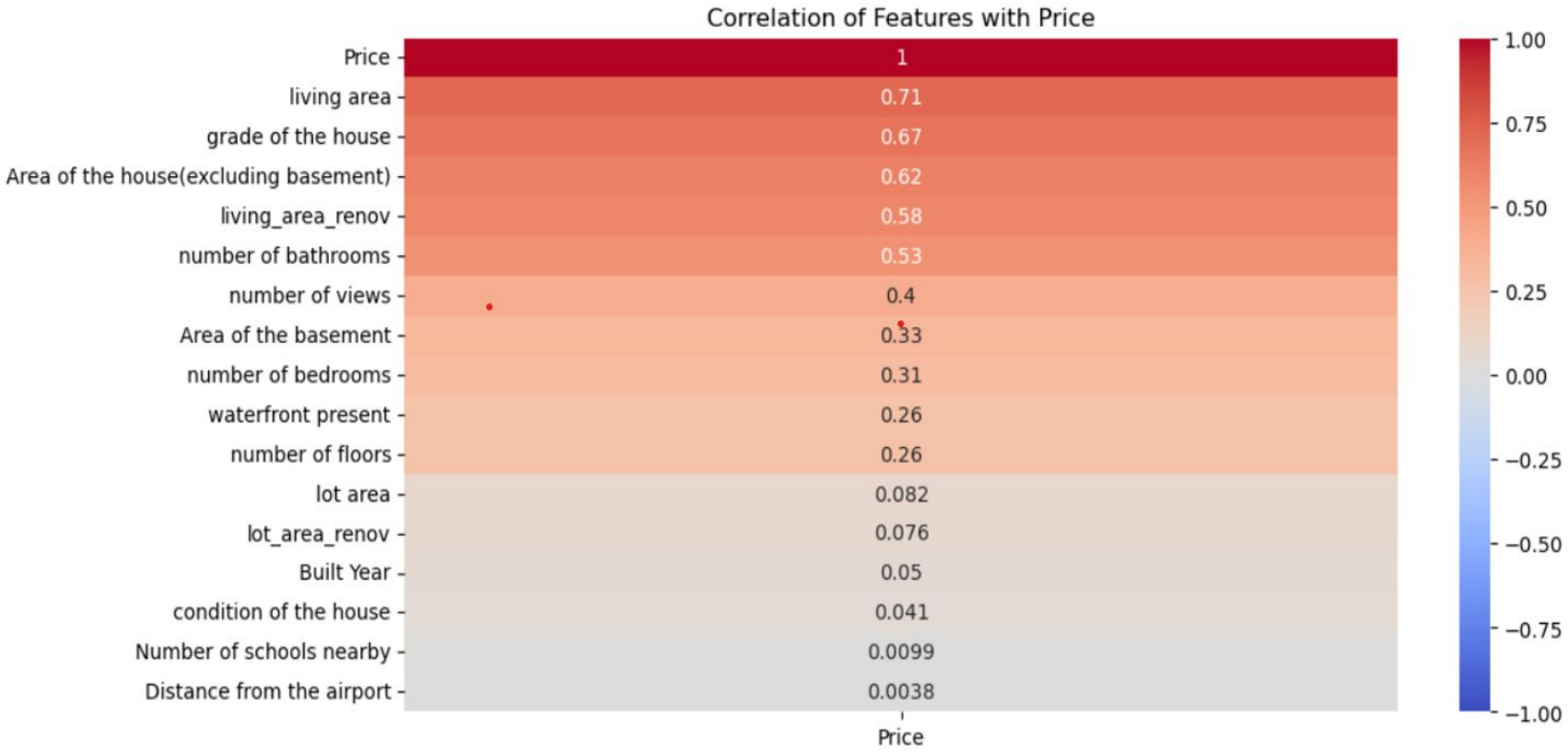


- https://data.world/dataindianset2000/house-price-india

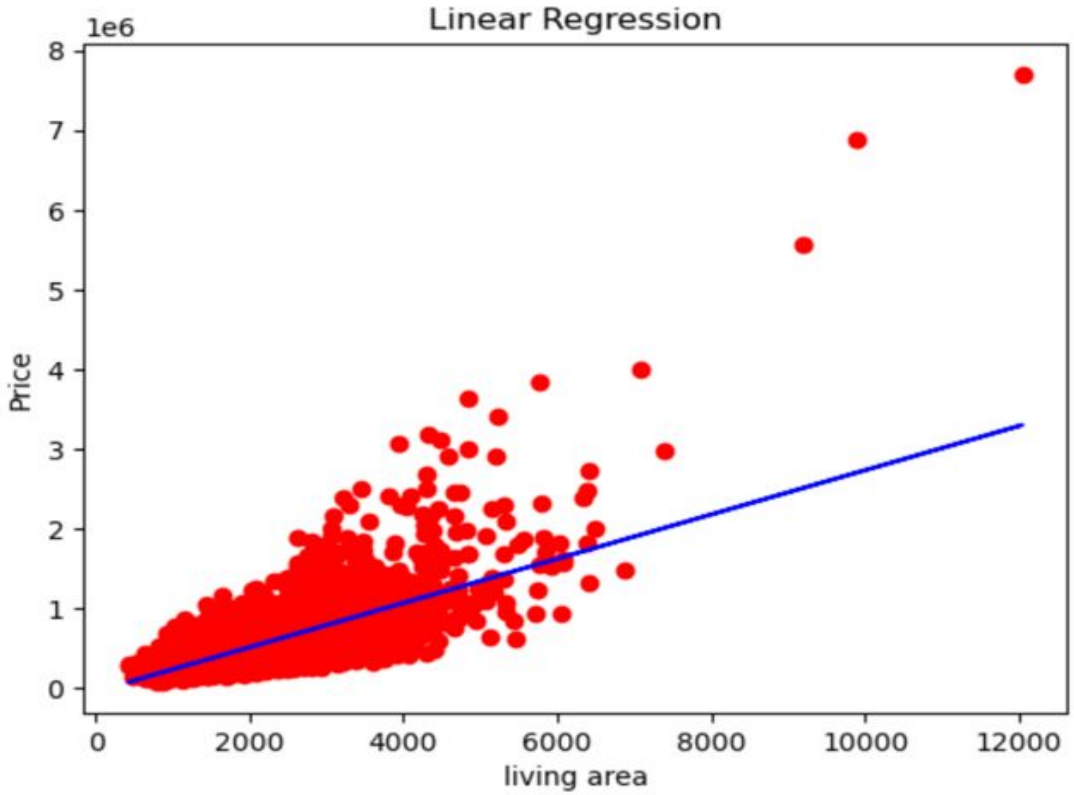
Dataset-Preprocessing

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14620 entries, 0 to 14619
Data columns (total 17 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   number of bedrooms                    14620 non-null  int64
 1   number of bathrooms                   14620 non-null  float64
 2   living area                           14620 non-null  int64
 3   lot area                              14620 non-null  int64
 4   number of floors                      14620 non-null  float64
 5   waterfront present                    14620 non-null  int64
 6   number of views                       14620 non-null  int64
 7   condition of the house                14620 non-null  int64
 8   grade of the house                    14620 non-null  int64
 9   Area of the house(excluding basement) 14620 non-null  int64
 10  Area of the basement                  14620 non-null  int64
 11  Built Year                            14620 non-null  int64
 12  living_area_renov                     14620 non-null  int64
 13  lot_area_renov                        14620 non-null  int64
 14  Number of schools nearby              14620 non-null  int64
 15  Distance from the airport             14620 non-null  int64
 16  Price                                 14620 non-null  int64
```

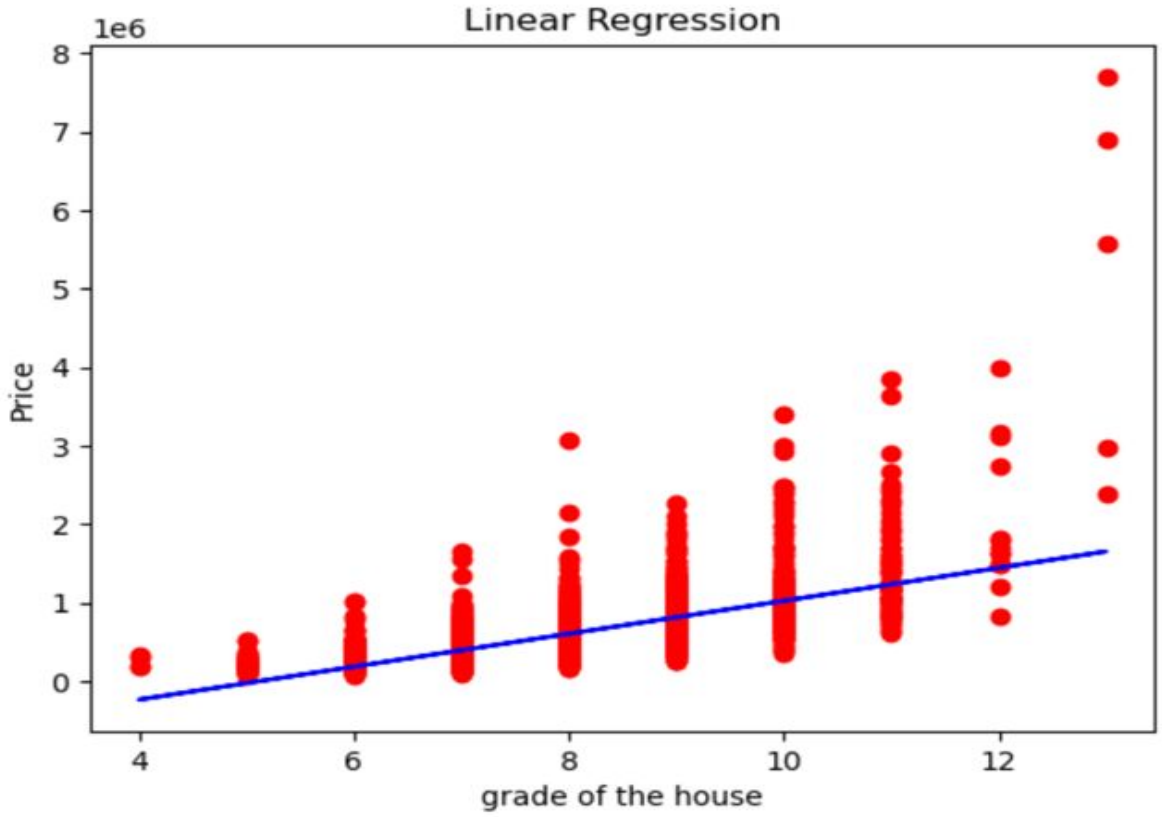
Correlation



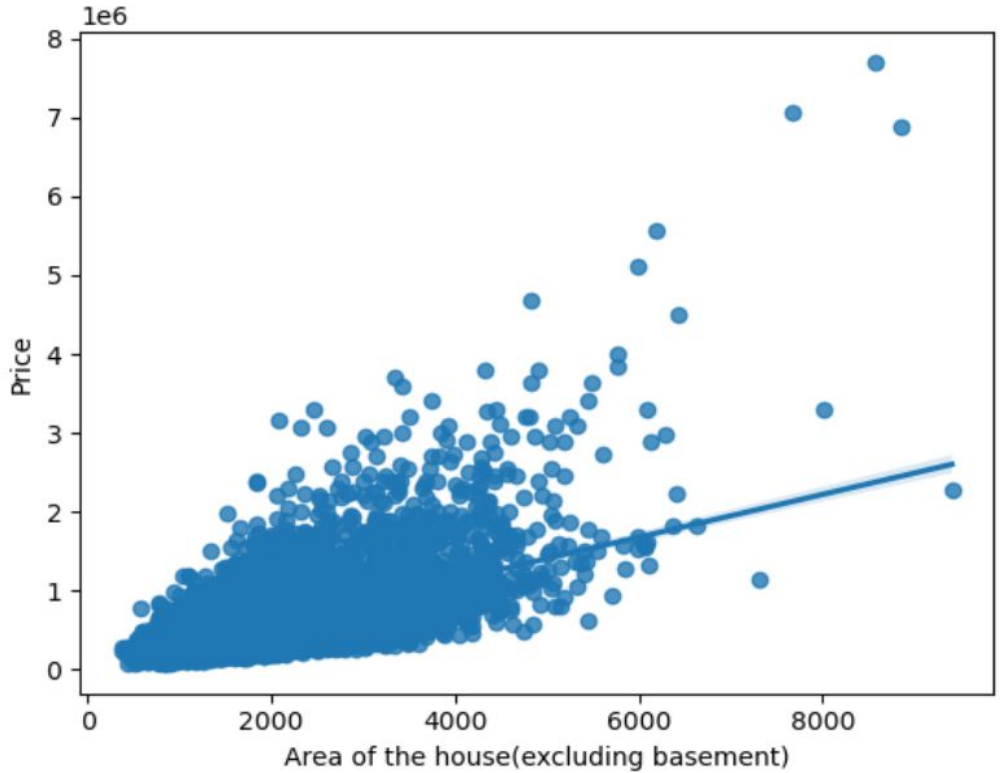
Regression plot(X = Living Area VS Y = Price)



Regression plot(X = Grade of House VS Y = Price)



Regression plot(X = Area of the House VS Y = Price)



Regression-Multiple Linear

Summary of Multiple Linear Regression

OLS Regression Results

Dep. Variable:	Price	R-squared:	0.664
Model:	OLS	Adj. R-squared:	0.664
Method:	Least Squares	F-statistic:	1926.
Date:	Wed, 02 Oct 2024	Prob (F-statistic):	0.00
Time:	10:20:48	Log-Likelihood:	-2.0012e+05
No. Observations:	14620	AIC:	4.003e+05
Df Residuals:	14604	BIC:	4.004e+05
Df Model:	15		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	✅ 6.156e+06	1.57e+05	39.153	0.000	5.85e+06	6.46e+06
number of bedrooms	-3.984e+04	2387.221	-16.688	0.000	-4.45e+04	-3.52e+04
number of bathrooms	4.652e+04	4141.671	11.232	0.000	3.84e+04	5.46e+04
living area	114.4892	2.876	39.814	0.000	108.853	120.126
lot area	-0.1311	0.066	-1.985	0.047	-0.261	-0.002
number of floors	2.474e+04	4536.891	5.454	0.000	1.59e+04	3.36e+04
waterfront present	5.77e+05	2.21e+04	26.078	0.000	5.34e+05	6.2e+05
number of views	4.217e+04	2736.267	15.410	0.000	3.68e+04	4.75e+04
condition of the house	2.151e+04	2921.274	7.364	0.000	1.58e+04	2.72e+04
grade of the house	1.189e+05	2712.828	43.834	0.000	1.14e+05	1.24e+05
Area of the house(excluding basement)	58.5015	2.827	20.691	0.000	52.959	64.043
Area of the basement	55.9878	3.313	16.901	0.000	49.495	62.481
Built Year	-3550.9701	80.317	-44.212	0.000	-3708.402	-3393.538
living_area_renov	13.0214	4.328	3.009	0.003	4.538	21.504
lot_area_renov	-0.5166	0.097	-5.331	0.000	-0.707	-0.327
Number of schools nearby	3040.0393	2157.430	1.409	0.159	-1188.796	7268.875
Distance from the airport	-113.8137	197.351	-0.577	0.564	-500.647	273.019

Omnibus:	11983.201	Durbin-Watson:	1.624
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1201714.919
Skew:	3.347	Prob(JB):	0.00
Kurtosis:	46.908	Cond. No.	2.95e+17

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 3.76e-22. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

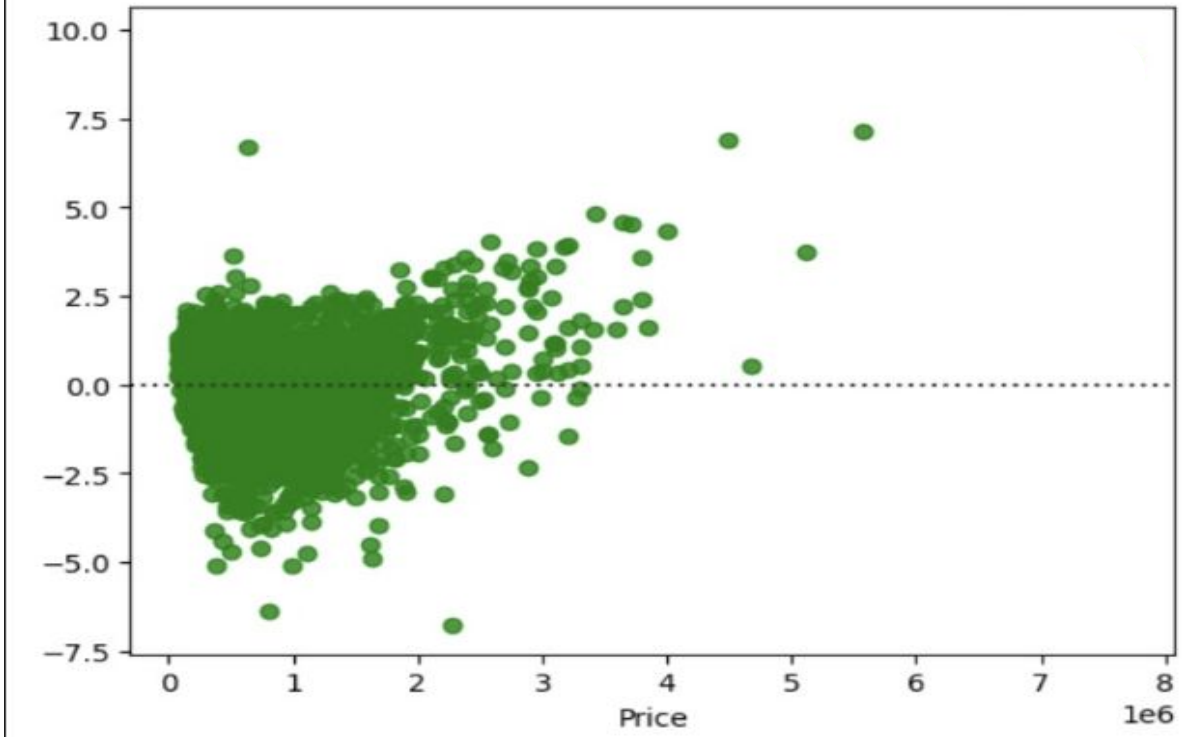
Error Matrix for Linear Regression

```
res= stat()
res.reg_metric(y=np.array(y), yhat=np.array(reg.predict(X)), resid=np.array(reg.resid))
res.reg_metric_df
```

	Metrics	Value
0	Root Mean Square Error (RMSE)	2.129635e+05
1	Mean Squared Error (MSE)	4.535346e+10
2	Mean Absolute Error (MAE)	1.366337e+05
3	Mean Absolute Percentage Error (MAPE)	2.851000e-01

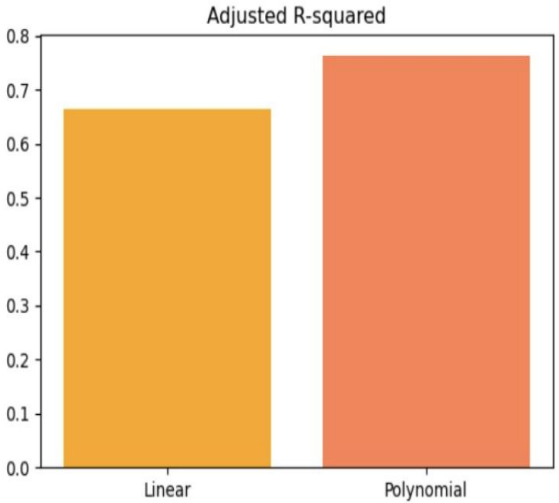
Residual plot

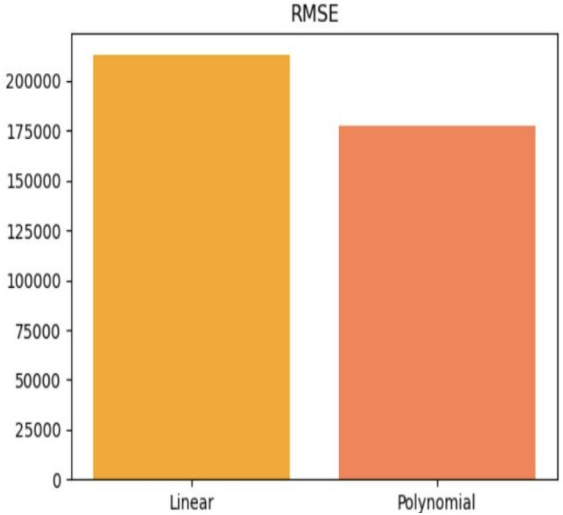
```
influence=reg.get_influence()
residual_price=influence.resid_studentized_external
sns.residplot(x = 'Price',y = residual_price,data = df,color='g');
```

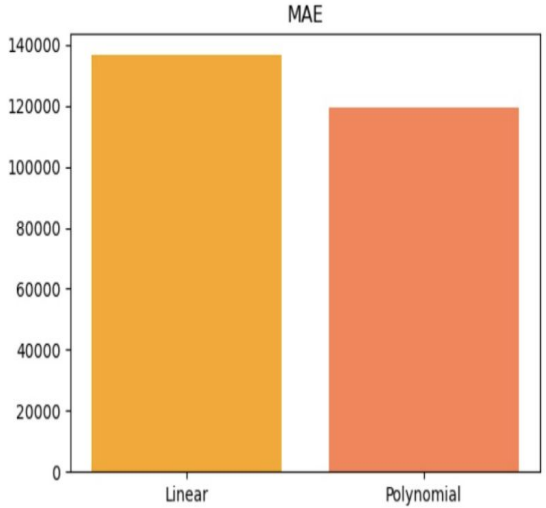


Regression-Polynomial

Comparing Multiple Linear and Polynomial Regression







Polynomial regression is the best regression model for the chosen dataset because the **AR-squared value is higher** for Poly-regression VS Linear and **RMSE and MAE are lower**.

Thank You!

scm516_group_project_final

October 5, 2024

[3]:
```
# Necessary imports for preprocessing and machine learning

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer   # interpolation for missing values
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import RocCurveDisplay, confusion_matrix, accuracy_score,␣
 ↪precision_score, recall_score, f1_score, classification_report
from imblearn.over_sampling import SMOTE
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
from bioinfokit.analys import stat
from bioinfokit import visuz
from sklearn.preprocessing import PolynomialFeatures
```

[28]:
```
# load the dataset

df_bank=pd.read_csv("bank-full.csv", delimiter=';')
```

[29]:
```
df_bank
```

[29]:

	age	job	marital	education	default	balance	housing	loan	\
0	58	management	married	tertiary	no	2143	yes	no	
1	44	technician	single	secondary	no	29	yes	no	
2	33	entrepreneur	married	secondary	no	2	yes	yes	
3	47	blue-collar	married	unknown	no	1506	yes	no	
4	33	unknown	single	unknown	no	1	no	no	
…	…	…	…	…	…	…	…	…	…

```
45206   51    technician   married   tertiary      no      825      no   no
45207   71       retired  divorced    primary      no     1729      no   no
45208   72       retired   married  secondary      no     5715      no   no
45209   57   blue-collar   married  secondary      no      668      no   no
45210   37  entrepreneur   married  secondary      no     2971      no   no

          contact  day month  duration  campaign  pdays  previous poutcome    y
0         unknown    5   may       261         1     -1         0  unknown   no
1         unknown    5   may       151         1     -1         0  unknown   no
2         unknown    5   may        76         1     -1         0  unknown   no
3         unknown    5   may        92         1     -1         0  unknown   no
4         unknown    5   may       198         1     -1         0  unknown   no
…           …   …   …       …         …    …        …       …  …
45206   cellular   17   nov       977         3     -1         0  unknown  yes
45207   cellular   17   nov       456         2     -1         0  unknown  yes
45208   cellular   17   nov      1127         5    184         3  success  yes
45209  telephone   17   nov       508         4     -1         0  unknown   no
45210   cellular   17   nov       361         2    188        11    other   no

[45211 rows x 17 columns]
```

[30]:
```
# Descriptive statistics of data

df_bank.describe()
```

[30]:
```
                age        balance           day      duration      campaign  \
count  45211.000000   45211.000000  45211.000000  45211.000000  45211.000000
mean      40.936210    1362.272058     15.806419    258.163080      2.763841
std       10.618762    3044.765829      8.322476    257.527812      3.098021
min       18.000000   -8019.000000      1.000000      0.000000      1.000000
25%       33.000000      72.000000      8.000000    103.000000      1.000000
50%       39.000000     448.000000     16.000000    180.000000      2.000000
75%       48.000000    1428.000000     21.000000    319.000000      3.000000
max       95.000000  102127.000000     31.000000   4918.000000     63.000000

              pdays      previous
count  45211.000000  45211.000000
mean      40.197828      0.580323
std      100.128746      2.303441
min       -1.000000      0.000000
25%       -1.000000      0.000000
50%       -1.000000      0.000000
75%       -1.000000      0.000000
max      871.000000    275.000000
```

[31]:
```
df_bank.drop(columns='contact', inplace=True)
df_bank.drop(columns='previous', inplace=True)
```

```
df_bank.drop(columns='poutcome', inplace=True)
df_bank.drop(columns='pdays', inplace=True)
df_bank.drop(columns='month', inplace=True)
df_bank.drop(columns='day', inplace=True)
```

[32]:
```
df_bank
```

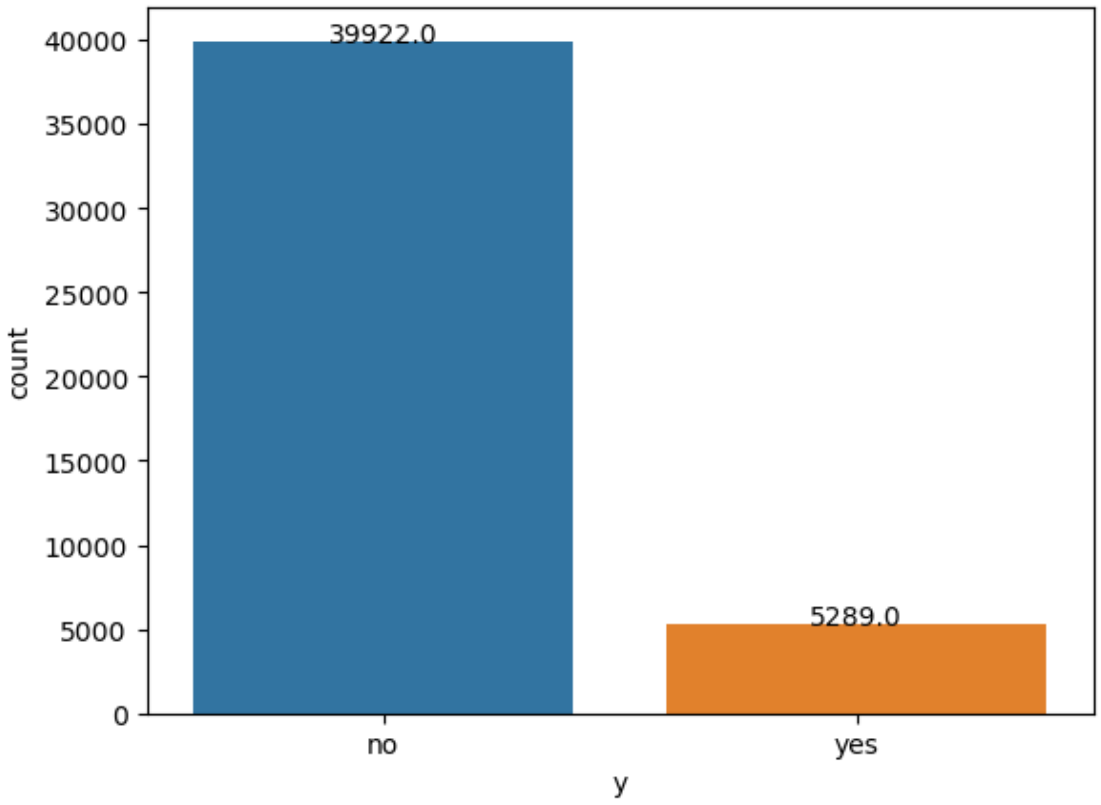
[32]:
```
       age           job   marital  education default  balance housing loan  \
0       58    management   married   tertiary      no     2143     yes   no
1       44    technician    single  secondary      no       29     yes   no
2       33  entrepreneur   married  secondary      no        2     yes  yes
3       47   blue-collar   married    unknown      no     1506     yes   no
4       33       unknown    single    unknown      no        1      no   no
…     …           …       …         …        …     …      …  …
45206   51    technician   married   tertiary      no      825      no   no
45207   71       retired  divorced    primary      no     1729      no   no
45208   72       retired   married  secondary      no     5715      no   no
45209   57   blue-collar   married  secondary      no      668      no   no
45210   37  entrepreneur   married  secondary      no     2971      no   no

       duration  campaign    y
0           261         1   no
1           151         1   no
2            76         1   no
3            92         1   no
4           198         1   no
…          …       …  …
45206       977         3  yes
45207       456         2  yes
45208      1127         5  yes
45209       508         4   no
45210       361         2   no

[45211 rows x 11 columns]
```

[33]:
```
# check the type of data and see if there is any missing value

df_bank.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 11 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        45211 non-null  int64
 1   job        45211 non-null  object
 2   marital    45211 non-null  object
 3   education  45211 non-null  object
```

```
 4   default    45211 non-null  object
 5   balance    45211 non-null  int64
 6   housing    45211 non-null  object
 7   loan       45211 non-null  object
 8   duration   45211 non-null  int64
 9   campaign   45211 non-null  int64
 10  y          45211 non-null  object
dtypes: int64(4), object(7)
memory usage: 3.8+ MB
```

[34]:
```
# Is the dataset balanced?

# Create the countplot
ax = sns.countplot(data=df_bank, x='y')

# Add count labels on top of the bars
for p in ax.patches:
    ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()),
                ha='center', va='baseline')

# Display the plot
plt.show()
```



[35]:
```
# Number of people with term_deposit
(df_bank[['y']]=="yes").sum()
```

[35]:
```
y    5289
dtype: int64
```

[36]:
```
# Number of people without term_deposit

(df_bank[['y']]=="no").sum()
```

[36]:
```
y    39922
dtype: int64
```

[37]:
```
# calculate probability of taking term_deposit

(df_bank[['y']]=="yes").sum() / len(df_bank['y'])
```

[37]:
```
y    0.116985
dtype: float64
```

[38]:
```
# How to get the list of all columns in my dataset

columns_list=df_bank.columns.tolist()
```

[39]:
```
columns_list
```

[39]:
```
['age',
 'job',
 'marital',
 'education',
 'default',
 'balance',
 'housing',
 'loan',
 'duration',
 'campaign',
 'y']
```

[40]:
```
# We need to create a variable that stores the name of categorical columns
# as well a varibale that stores the name of numerical variable.
# because we need to handle missing values for categorical using mode technique
# we also need to replace the missing values of numerical variable using mean.␣
 ↪You can alos use linear interpolcation

# List of categorical variables (dtype == object or category)
```

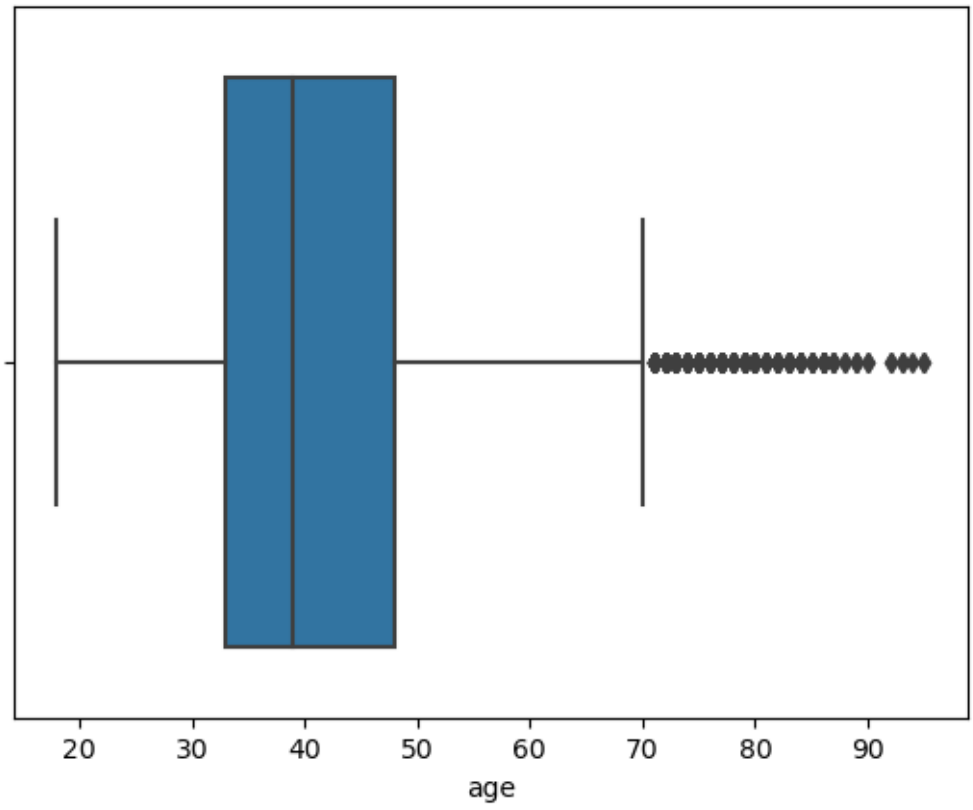
```
categorical_cols = df_bank.select_dtypes(include=['object', 'category']).
 ↪columns.tolist()

# List of numerical variables (dtype == int or float)
numerical_cols = df_bank.select_dtypes(include=['number']).columns.tolist()

# Display the lists
print("Categorical columns:", categorical_cols)
print("Numerical columns:", numerical_cols)
```

```
Categorical columns: ['job', 'marital', 'education', 'default', 'housing',
'loan', 'y']
Numerical columns: ['age', 'balance', 'duration', 'campaign']
```

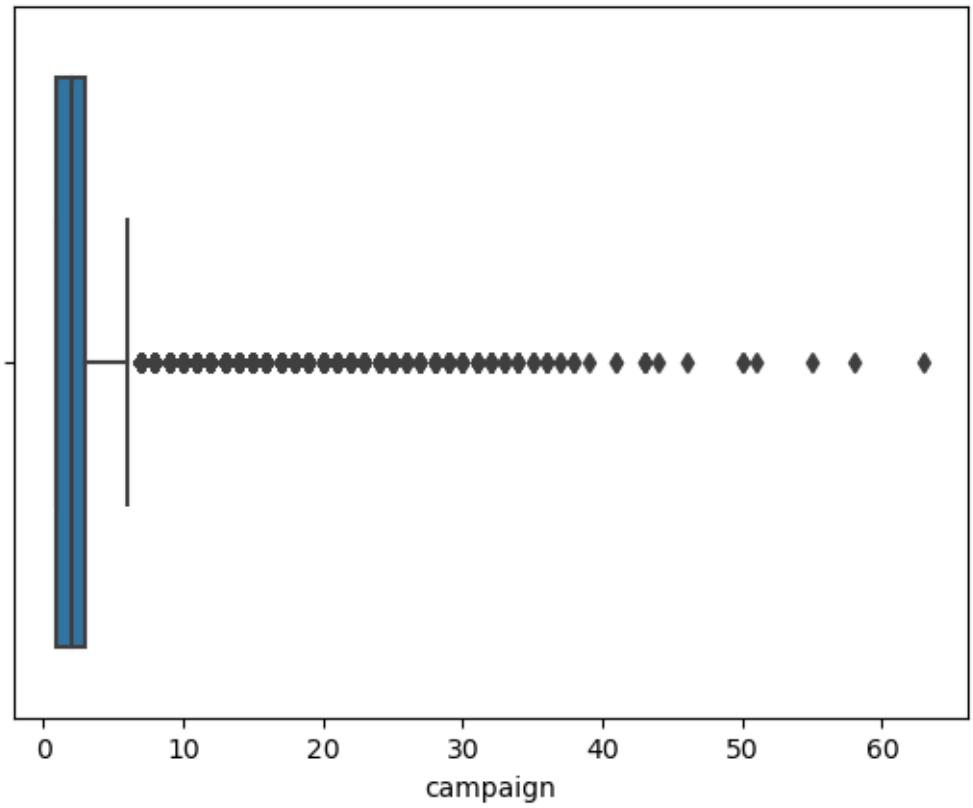
[41]:
```
df_bank.isnull().sum()
```

[41]:
```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
duration     0
campaign     0
y            0
dtype: int64
```

[42]:
```
# Find and remove outlier from numerical variable

# Create a boxplot for the 'age' column to detect outliers
sns.boxplot(data=df_bank, x='age');    # there are  outliers (black dots). Thos␣
 ↪outliers should be removed using IQR method
```



[43]:
```
sns.boxplot(data=df_bank, x='campaign');
```



[44]:
```
# Calculate the IQR for the 'campaign' column
Q1 = df_bank['campaign'].quantile(0.25) # 25th percentile
Q3 = df_bank['campaign'].quantile(0.75) # 75th percentile
IQR = Q3 - Q1 # Interquartile range
# Define the lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
# Filter out the outliers
df_bank = df_bank[(df_bank['campaign'] >= lower_bound) & (df_bank['campaign']␣
 ↪<= upper_bound)]
# df1 is a new datafram after removing outliers
# Display the results (rows without outliers)
```

[45]:
```
df_bank
```

[45]:

	age	job	marital	education	default	balance	housing	loan
0	58	management	married	tertiary	no	2143	yes	no
1	44	technician	single	secondary	no	29	yes	no
2	33	entrepreneur	married	secondary	no	2	yes	yes
3	47	blue-collar	married	unknown	no	1506	yes	no
4	33	unknown	single	unknown	no	1	no	no

```
…    …            …        …         …      …       …      …  …
45206   51    technician   married   tertiary      no      825      no   no
45207   71       retired  divorced    primary      no     1729      no   no
45208   72       retired   married  secondary      no     5715      no   no
45209   57   blue-collar   married  secondary      no      668      no   no
45210   37  entrepreneur   married  secondary      no     2971      no   no

       duration  campaign    y
0           261         1   no
1           151         1   no
2            76         1   no
3            92         1   no
4           198         1   no
…           …         …  …
45206       977         3  yes
45207       456         2  yes
45208      1127         5  yes
45209       508         4   no
45210       361         2   no

[42147 rows x 11 columns]
```

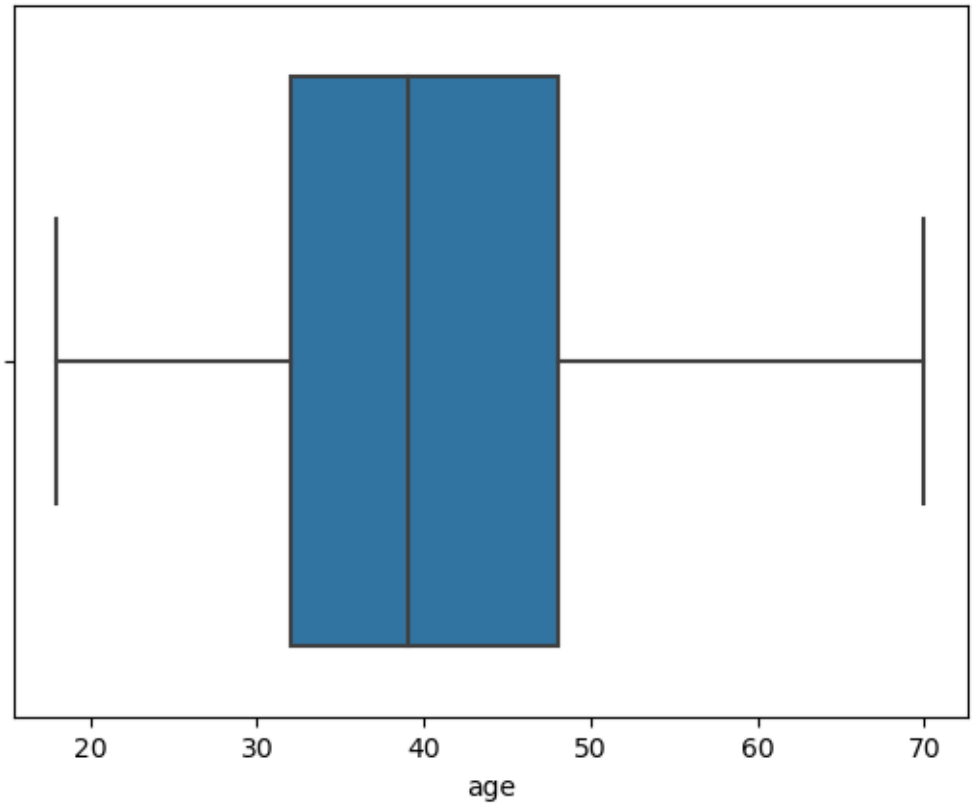
[46]:
```
# Calculate the IQR for the 'age' column
Q1 = df_bank['age'].quantile(0.25) # 25th percentile
Q3 = df_bank['age'].quantile(0.75) # 75th percentile
IQR = Q3 - Q1 # Interquartile range
# Define the lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
# Filter out the outliers
df_bank = df_bank[(df_bank['age'] >= lower_bound) & (df_bank['age'] <=␣
 ↪upper_bound)]
```

[47]:
```
df_bank
```

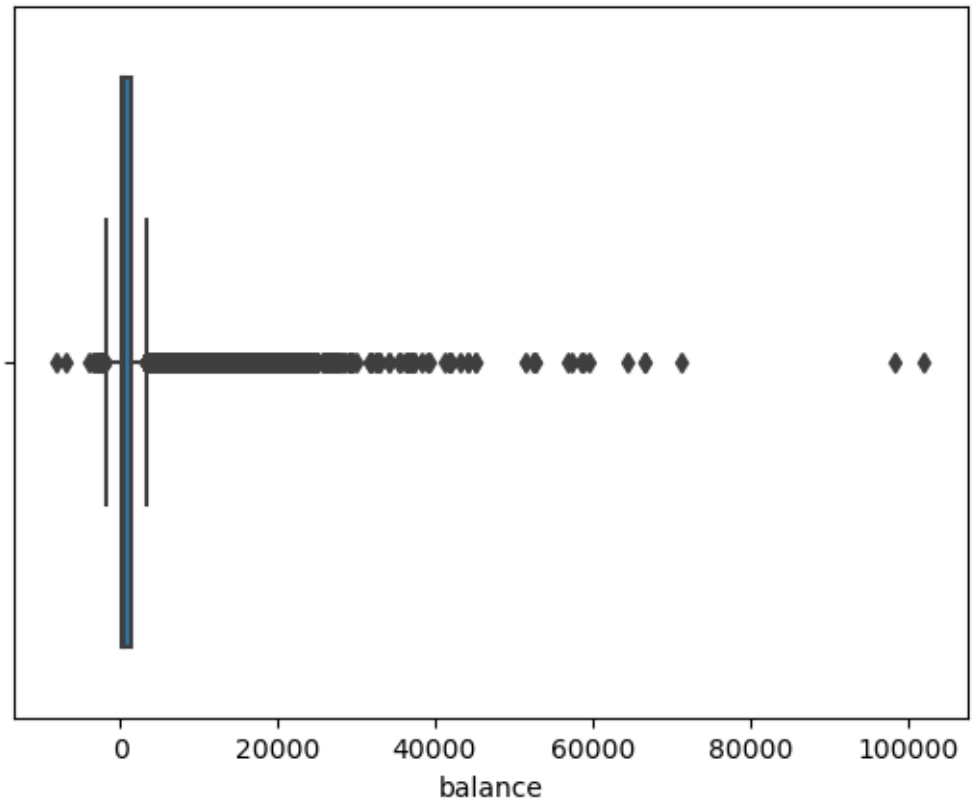
[47]:
```
       age           job  marital  education default  balance housing loan  \
0       58    management  married   tertiary      no     2143     yes   no
1       44    technician   single  secondary      no       29     yes   no
2       33  entrepreneur  married  secondary      no        2     yes  yes
3       47   blue-collar  married    unknown      no     1506     yes   no
4       33       unknown   single    unknown      no        1      no   no
…    …           …       …        …      …       …      …  …
45203   23       student   single   tertiary      no      113      no   no
45205   25    technician   single  secondary      no      505      no  yes
45206   51    technician  married   tertiary      no      825      no   no
45209   57   blue-collar  married  secondary      no      668      no   no
45210   37  entrepreneur  married  secondary      no     2971      no   no
```

```
       duration  campaign    y
0           261         1   no
1           151         1   no
2            76         1   no
3            92         1   no
4           198         1   no
…          …       …   …
45203       266         1  yes
45205       386         2  yes
45206       977         3  yes
45209       508         4   no
45210       361         2   no

[41674 rows x 11 columns]
```

[48]:
```
sns.boxplot(data=df_bank, x='age');
```



[49]:
```
sns.boxplot(data=df_bank, x='balance');
```



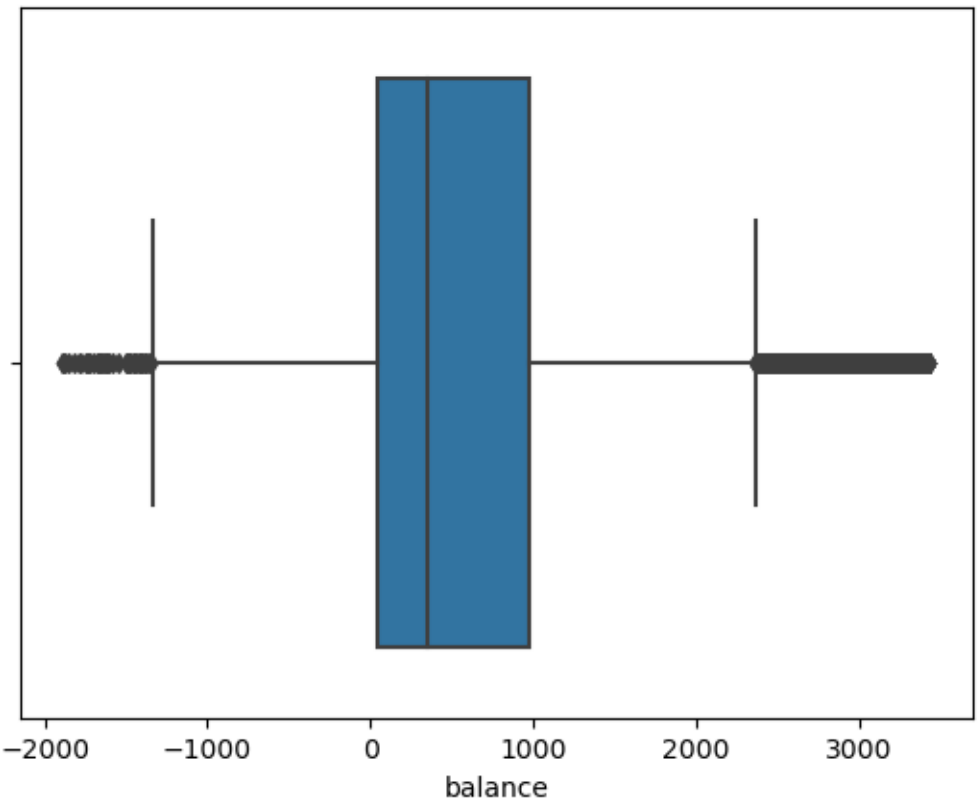
[50]:
```
# Calculate the IQR for the 'balance' column
Q1 = df_bank['balance'].quantile(0.25) # 25th percentile
Q3 = df_bank['balance'].quantile(0.75) # 75th percentile
IQR = Q3 - Q1 # Interquartile range
# Define the lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
# Filter out the outliers
df_bank = df_bank[(df_bank['balance'] >= lower_bound) & (df_bank['balance'] <=␣
 ↪upper_bound)]
df_bank
```

[50]:

	age	job	marital	education	default	balance	housing	loan	\
0	58	management	married	tertiary	no	2143	yes	no	
1	44	technician	single	secondary	no	29	yes	no	
2	33	entrepreneur	married	secondary	no	2	yes	yes	
3	47	blue-collar	married	unknown	no	1506	yes	no	
4	33	unknown	single	unknown	no	1	no	no	
…	…	…	…	…	…	…	…	…	…
45203	23	student	single	tertiary	no	113	no	no	
45205	25	technician	single	secondary	no	505	no	yes	

```
45206   51    technician  married   tertiary      no      825      no   no
45209   57   blue-collar  married  secondary      no      668      no   no
45210   37  entrepreneur  married  secondary      no     2971      no   no

       duration  campaign    y
0           261         1   no
1           151         1   no
2            76         1   no
3            92         1   no
4           198         1   no
…           …           …   …
45203       266         1  yes
45205       386         2  yes
45206       977         3  yes
45209       508         4   no
45210       361         2   no

[37303 rows x 11 columns]
```

[51]:
```
sns.boxplot(data=df_bank, x='balance');
```



[52]:
```
df_bank
```

[52]:
```
       age           job  marital  education default  balance housing loan  \
0       58    management  married   tertiary      no     2143     yes   no
1       44    technician   single  secondary      no       29     yes   no
2       33  entrepreneur  married  secondary      no        2     yes  yes
3       47   blue-collar  married    unknown      no     1506     yes   no
4       33       unknown   single    unknown      no        1      no   no
…      …             …       …         …      …      …   …
45203   23       student   single   tertiary      no      113      no   no
45205   25    technician   single  secondary      no      505      no  yes
45206   51    technician  married   tertiary      no      825      no   no
45209   57   blue-collar  married  secondary      no      668      no   no
45210   37  entrepreneur  married  secondary      no     2971      no   no

       duration  campaign    y
0           261         1   no
1           151         1   no
2            76         1   no
3            92         1   no
4           198         1   no
…          …       …  …
45203       266         1  yes
45205       386         2  yes
45206       977         3  yes
45209       508         4   no
45210       361         2   no

[37303 rows x 11 columns]
```

[53]:
```
# label Encoding
# we use the following function to encode categorical variable

from sklearn.preprocessing import LabelEncoder

lb=LabelEncoder()

for col in categorical_cols:    # we define categorical_cols   in line [73]
    df_bank.loc[:,col]=lb.fit_transform(df_bank[col])   # note that df1 is new␣
 ↪dataframe without outliers
```

[54]:
```
df_bank.head(5)
```

[54]:
```
   age job marital education default  balance housing loan  duration  \
0   58   4       1         2       0     2143       1    0       261
1   44   9       2         1       0       29       1    0       151
2   33   2       1         1       0        2       1    1        76
```

```
3   47   1        1          3        0     1506        1    0         92
4   33  11        2          3        0        1        0    0        198

   campaign  y
0         1  0
1         1  0
2         1  0
3         1  0
4         1  0
```

[55]:
```
# Standardize the numerical variable (Age variable)

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

df_bank.loc[:,numerical_cols] = scaler.fit_transform(df_bank[numerical_cols])
 ↪# we define list of numerical variable in line [73]
```

[56]:
```
df_bank
```

[56]:
```
             age job marital education default   balance housing loan  \
0       0.769231   4       1         2       0  0.757952       1    0
1       0.500000   9       2         1       0  0.360060       1    0
2       0.288462   2       1         1       0  0.354978       1    1
3       0.557692   1       1         3       0  0.638058       1    0
4       0.288462  11       2         3       0  0.354790       0    0
…            …   ..     …         …     …         …     …  …
45203   0.096154   8       2         2       0  0.375871       0    0
45205   0.134615   9       2         1       0  0.449652       0    1
45206   0.634615   9       1         2       0  0.509881       0    0
45209   0.750000   1       1         1       0  0.480331       0    0
45210   0.365385   2       1         1       0  0.913796       0    0

       duration  campaign  y
0      0.067251       0.0  0
1      0.038907       0.0  0
2      0.019583       0.0  0
3      0.023705       0.0  0
4      0.051018       0.0  0
…           …       … ..
45203  0.068539       0.0  1
45205  0.099459       0.2  1
45206  0.251739       0.4  1
45209  0.130894       0.6  0
45210  0.093017       0.2  0
```

```
[37303 rows x 11 columns]
```

[57]:
```
# define features (X) and response/label (y)

X= df_bank.drop(columns=['y'])
y=df_bank['y']
```

[39]:
```
# We saw earlier that the data is not balanced. We apply SMOTE sampling␣
 ↪technique to our imbalanced data
smote = SMOTE()
X_res, y_res = smote.fit_resample(X,y.astype(int))
```

[40]:
```
y_res
```

[40]:
```
0        0
1        0
2        0
3        0
4        0
        ..
66169    1
66170    1
66171    1
66172    1
66173    1
Name: y, Length: 66174, dtype: int32
```

[41]:
```
y_res.value_counts()  # the response/ label has been balanced
```

[41]:
```
y
0    33087
1    33087
Name: count, dtype: int64
```

[42]:
```
# split into training and testing test

X_train, X_test, y_train, y_test = train_test_split(X_res, y_res, test_size=0.
 ↪3, random_state=8)
```

[43]:
```
# Naive Bayes Model

nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
y_pred_nb = nb_model.predict(X_test)
```

[44]:
```
# Decision Tree Model
```

```
dt_model = DecisionTreeClassifier()
dt_model.fit(X_train, y_train)
y_pred_dt = dt_model.predict(X_test)
```

[45]:
```
# Random Forest Model

rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)
y_pred_rf=rf_model.predict(X_test)
```

[46]:
```
# KNN model

knn_model =KNeighborsClassifier(n_neighbors=3)
knn_model.fit(X_train, y_train)
y_pred_knn = knn_model.predict(X_test)
```

[47]:
```
# Evaluation: Confusion Matrix and Metrics

def evaluate_model(y_test, y_pred, model_name):
    print(f"Model: {model_name}")
    print("Confusion Matrix:")
    print(confusion_matrix(y_test, y_pred))
    print(f"Accuracy:  {accuracy_score(y_test, y_pred):.2f}")
    print(f"Precision: {precision_score(y_test, y_pred):.2f}")
    print(f"PRecall:   {recall_score(y_test, y_pred):.2f}")
    print(f"F1 Score:  {f1_score(y_test, y_pred):.2f}\n")
```

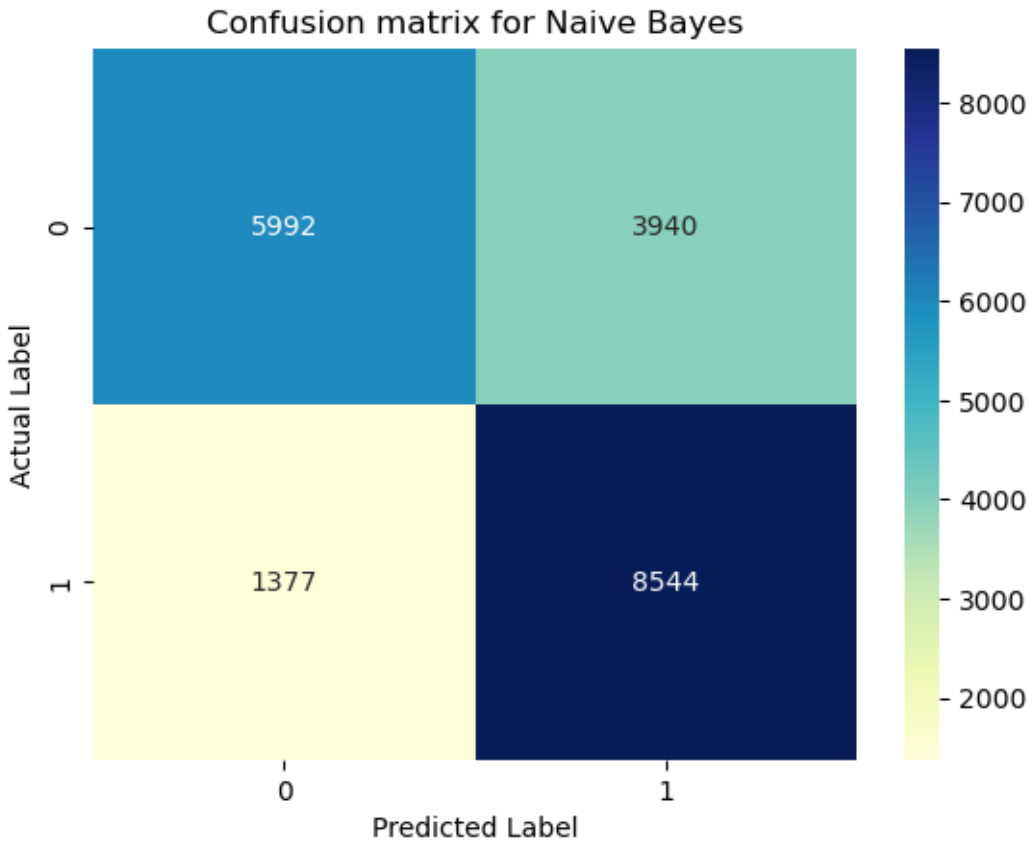
[48]:
```
# Evaluate all models

evaluate_model(y_test, y_pred_nb, "Naive Bayes")
evaluate_model(y_test, y_pred_dt, "Decision Tree")
evaluate_model(y_test, y_pred_rf, "Random Forest")
evaluate_model(y_test, y_pred_knn, "KNN")
```

```
Model: Naive Bayes
Confusion Matrix:
[[5992 3940]
 [1377 8544]]
Accuracy:  0.73
Precision: 0.68
PRecall:   0.86
F1 Score:  0.76

Model: Decision Tree
Confusion Matrix:
[[8160 1772]
 [1630 8291]]
Accuracy:  0.83
```

```
Precision: 0.82
PRecall:   0.84
F1 Score:  0.83

Model: Random Forest
Confusion Matrix:
[[8569 1363]
 [ 764 9157]]
Accuracy:  0.89
Precision: 0.87
PRecall:   0.92
F1 Score:  0.90

Model: KNN
Confusion Matrix:
[[8056 1876]
 [ 526 9395]]
Accuracy:  0.88
Precision: 0.83
PRecall:   0.95
F1 Score:  0.89
```

[49]:
```
cm1=confusion_matrix(y_test, y_pred_nb)
cm2=confusion_matrix(y_test, y_pred_dt)
cm3=confusion_matrix(y_test, y_pred_rf)
cm4=confusion_matrix(y_test, y_pred_knn)
```

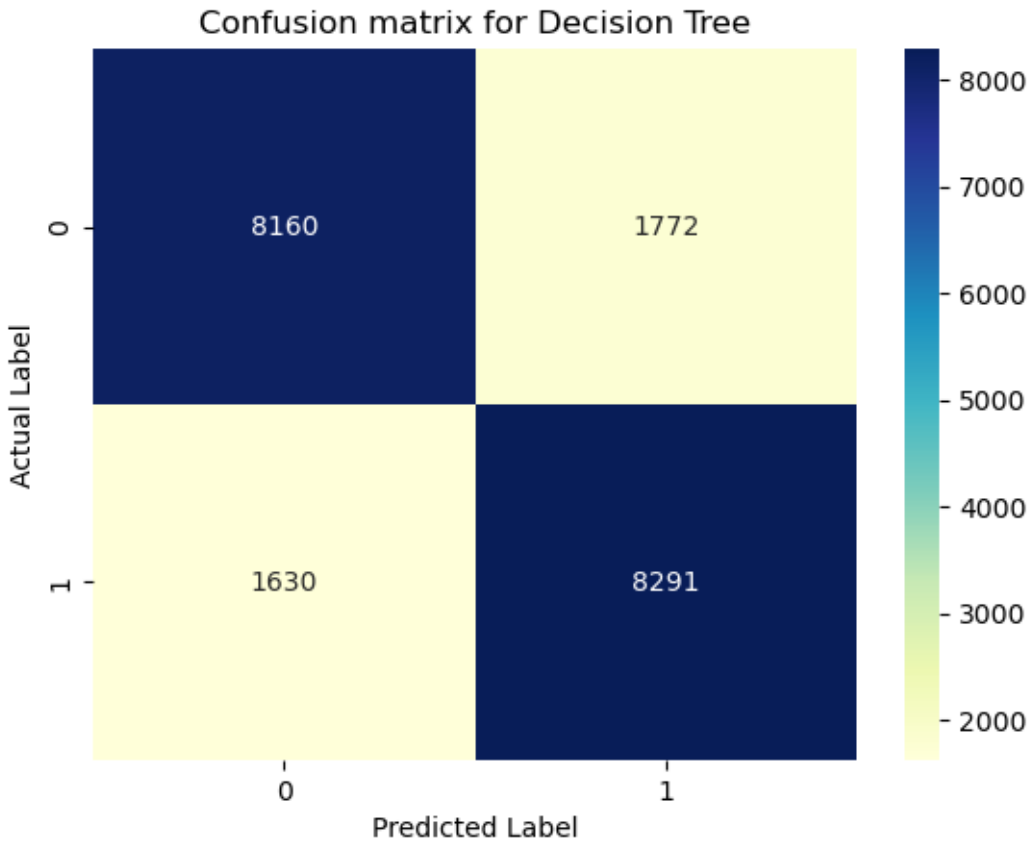
[50]:
```
# confusion matrix for Naive Bayes

sns.heatmap(cm1, annot=True, cmap='YlGnBu', fmt='g')
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')
plt.title('Confusion matrix for Naive Bayes');
```



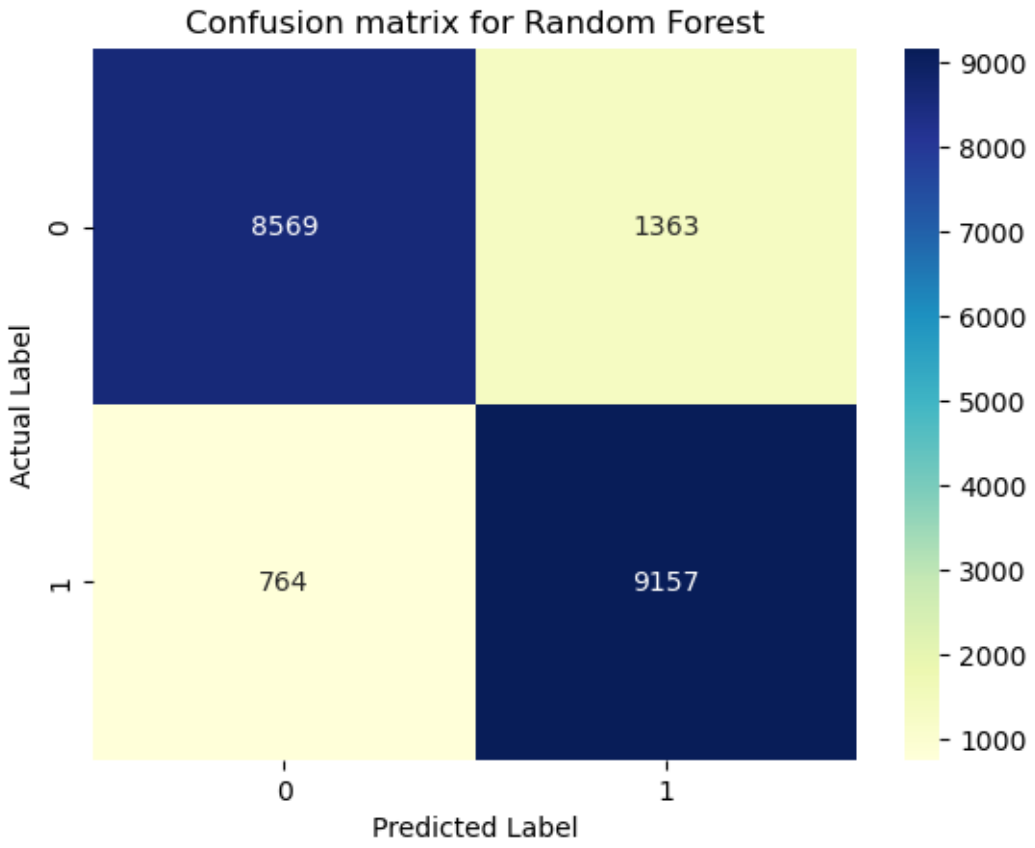
[51]:
```
print ("Accuracy of Naive Bayes for training datatse is:", nb_model.
↪score(X_train, y_train))
```

```
Accuracy of Naive Bayes for training datatse is: 0.7413052395241899
```

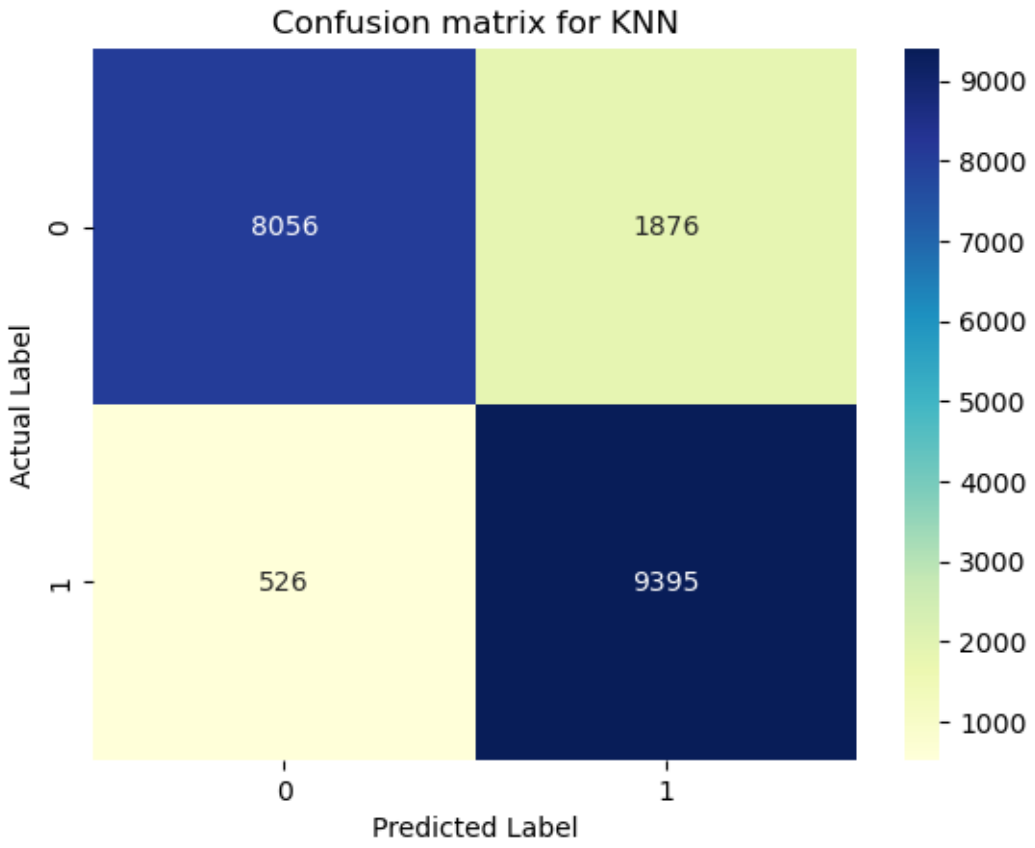
[54]:
```
# confusion matrix for Decision Tree

sns.heatmap(cm2, annot=True, cmap='YlGnBu', fmt='g')
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')
plt.title('Confusion matrix for Decision Tree');
```



[55]:
```
# confusion matrix for Raqndom Forest

sns.heatmap(cm3, annot=True, cmap='YlGnBu', fmt='g')
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')
plt.title('Confusion matrix for Random Forest');
```



[56]:
```
# confusion matrix for KNN

sns.heatmap(cm4, annot=True, cmap='YlGnBu', fmt='g')
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')
plt.title('Confusion matrix for KNN');
```

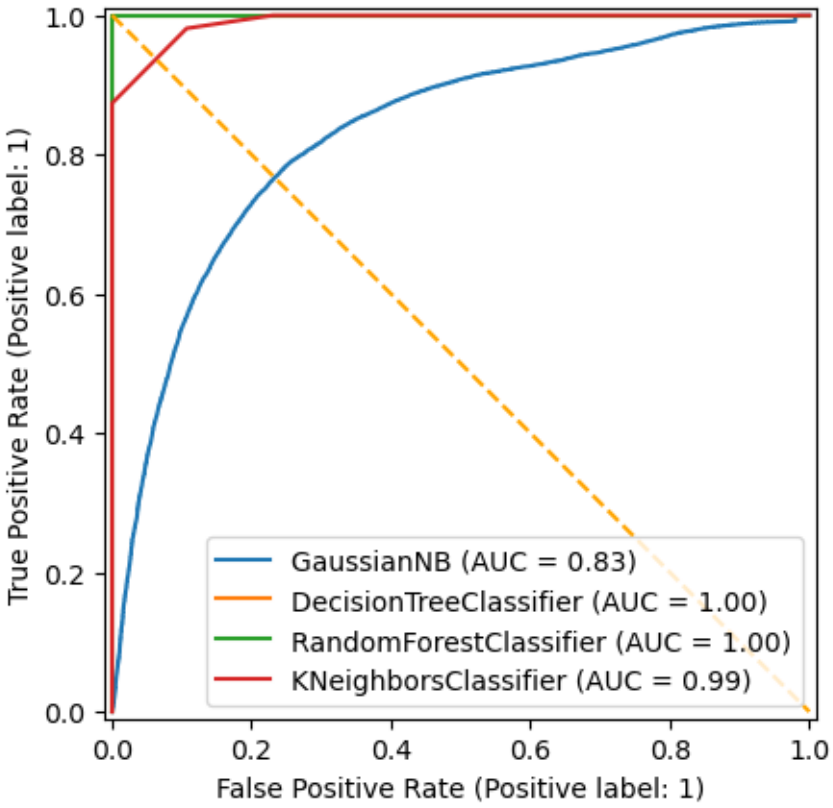


[57]:
```
# Random forest has the best performance. let's all the measures in a report
print(classification_report(y_test, y_pred_rf))
```

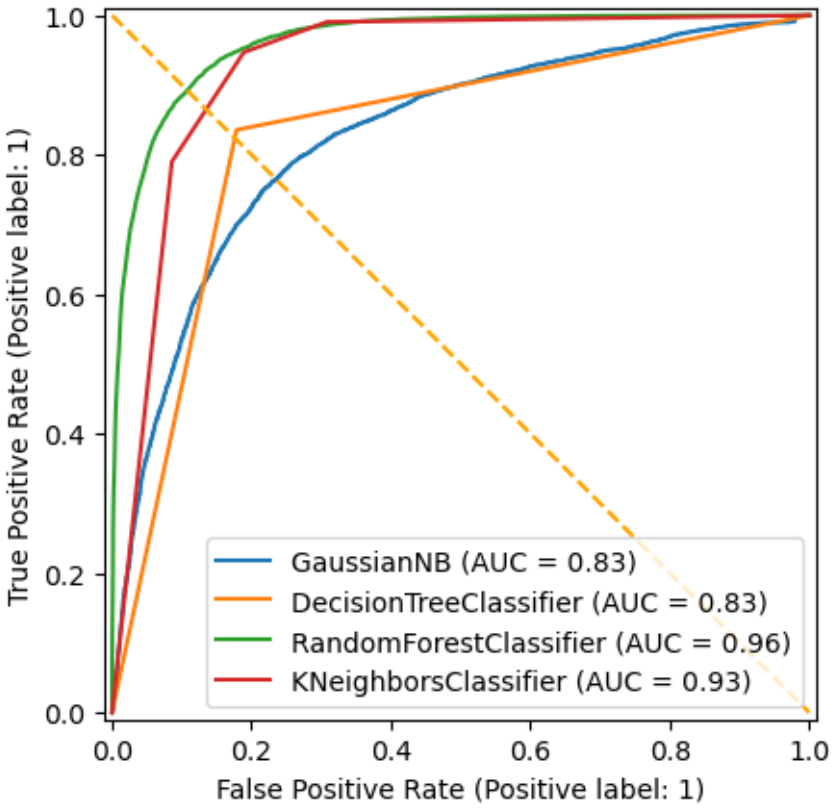
```
              precision    recall  f1-score   support

           0       0.92      0.86      0.89      9932
           1       0.87      0.92      0.90      9921

    accuracy                           0.89     19853
   macro avg       0.89      0.89      0.89     19853
weighted avg       0.89      0.89      0.89     19853
```

[58]:
```
# plot ROC curve for Xtrain and Xtest

from sklearn.metrics import RocCurveDisplay

disp =RocCurveDisplay.from_estimator(nb_model, X_train, y_train)
RocCurveDisplay.from_estimator(dt_model, X_train, y_train, ax=disp.ax_)
RocCurveDisplay.from_estimator(rf_model, X_train, y_train, ax=disp.ax_)
```

```
RocCurveDisplay.from_estimator(knn_model, X_train, y_train, ax=disp.ax_)
plt.plot([0,1], [1,0], color='orange', linestyle='--');
```



[59]:
```
# ROC Curve for Testing dataset

disp =RocCurveDisplay.from_estimator(nb_model, X_test, y_test)
RocCurveDisplay.from_estimator(dt_model, X_test, y_test, ax=disp.ax_)
RocCurveDisplay.from_estimator(rf_model, X_test, y_test, ax=disp.ax_)
RocCurveDisplay.from_estimator(knn_model, X_test, y_test, ax=disp.ax_)
plt.plot([0,1], [1,0], color='orange', linestyle='--');
```



[60]:
```
# Make a new prediction for new clients (new observations)

# load new observations


# Define the data
data = {
    'age': [35, 28, 52, 43],
    'job': ['teacher', 'artist', 'entrepreneur', 'engineer'],
    'marital': ['single', 'divorced', 'married', 'married'],
    'education': ['tertiary', 'secondary', 'tertiary', 'secondary'],
    'default': ['no', 'no', 'yes', 'no'],
    'balance': [1200, 850, 2100, 3500],
    'housing': ['yes', 'no', 'yes', 'no'],
    'loan': ['no', 'yes', 'yes', 'no'],
    'contact': ['cellular', 'cellular', 'unknown', 'telephone'],
    'day': [12, 24, 17, 5],
    'month': ['jul', 'sep', 'aug', 'jun'],
    'duration': [320, 145, 180, 400],
    'campaign': [3, 2, 1, 4],
    'pdays': [45, -1, 60, 30],
```

```
    'previous': [1, 0, 2, 1],
    'poutcome': ['success', 'unknown', 'success', 'failure'],
    'y': ['yes', 'no', 'yes', 'no']
}

# Create the DataFrame
df_sample = pd.DataFrame(data)

# Display the DataFrame
df_sample
```

[60]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	35	teacher	single	tertiary	no	1200	yes	no	cellular	12	jul	320	3	45	1	success	yes
1	28	artist	divorced	secondary	no	850	no	yes	cellular	24	sep	145	2	-1	0	unknown	no
2	52	entrepreneur	married	tertiary	yes	2100	yes	yes	unknown	17	aug	180	1	60	2	success	yes
3	43	engineer	married	secondary	no	3500	no	no	telephone	5	jun	400	4	30	1	failure	no

[61]:
```
df_sample
```

[61]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	35	teacher	single	tertiary	no	1200	yes	no	cellular	12	jul	320	3	45	1	success	yes
1	28	artist	divorced	secondary	no	850	no	yes	cellular	24	sep	145	2	-1	0	unknown	no
2	52	entrepreneur	married	tertiary	yes	2100	yes	yes	unknown	17	aug	180	1	60	2	success	yes
3	43	engineer	married	secondary	no	3500	no	no	telephone	5	jun	400	4	30	1	failure	no

[62]:
```
df_sample.drop(columns='contact', inplace=True)
df_sample.drop(columns='previous', inplace=True)
df_sample.drop(columns='poutcome', inplace=True)
df_sample.drop(columns='pdays', inplace=True)
df_sample.drop(columns='month', inplace=True)
df_sample.drop(columns='day', inplace=True)
```

[63]:
```
df_sample
```

[63]:
```
   age           job   marital  education default  balance housing loan  \
0   35       teacher    single   tertiary      no     1200     yes   no
1   28        artist  divorced  secondary      no      850      no  yes
2   52  entrepreneur   married   tertiary     yes     2100     yes  yes
3   43      engineer   married  secondary      no     3500      no   no

   duration  campaign    y
0       320         3  yes
1       145         2   no
2       180         1  yes
3       400         4   no
```

1

[64]:
```
lb=LabelEncoder()

for col in categorical_cols:
    df_sample.loc[:,col]=lb.fit_transform(df_sample[col])
scaler = MinMaxScaler()

df_sample.loc[:,numerical_cols] = scaler.
 ↪fit_transform(df_sample[numerical_cols])
```

[65]:
```
df_sample
```

[65]:
```
        age job marital education default   balance housing loan  duration  \
0  0.291667   3       2         1       0  0.132075       1    0  0.686275
1  0.000000   0       0         0       0  0.000000       0    1  0.000000
2  1.000000   2       1         1       1  0.471698       1    1  0.137255
3  0.625000   1       1         0       0  1.000000       0    0  1.000000

   campaign  y
0  0.666667  1
1  0.333333  0
2  0.000000  1
3  1.000000  0
```

[66]:
```
df_sample.drop(columns='y',inplace = True)
```

[67]:
```
df_sample
```

[67]:
```
        age job marital education default   balance housing loan  duration  \
0  0.291667   3       2         1       0  0.132075       1    0  0.686275
1  0.000000   0       0         0       0  0.000000       0    1  0.000000
2  1.000000   2       1         1       1  0.471698       1    1  0.137255
3  0.625000   1       1         0       0  1.000000       0    0  1.000000
```

```
   campaign
0  0.666667
1  0.333333
2  0.000000
3  1.000000
```

[68]:
```
y_pred_new_observation_rf=rf_model.predict(df_sample)
```

[69]:
```
y_pred_new_observation_nb = nb_model.predict(df_sample)
```

[70]:
```
y_pred_new_observation_dt = dt_model.predict(df_sample)
```

[71]:
```
y_pred_new_observation_knn = knn_model.predict(df_sample)
```

[72]:
```
y_pred_new_observation_dt
```

[72]:
```
array([0, 0, 0, 0])
```

[73]:
```
y_pred_new_observation_knn
```

[73]:
```
array([0, 0, 0, 1])
```

[74]:
```
y_pred_new_observation_nb
```

[74]:
```
array([1, 0, 0, 1])
```

[75]:
```
y_pred_new_observation_rf
```

[75]:
```
array([0, 0, 1, 1])
```

1.0.1 Feature Importance

[76]:
```
# Find the importance of features ----- Features importance using Random Forest

rf=RandomForestClassifier(random_state=1, n_estimators=100)
```

[77]:
```
y
```

[77]:
```
0        0
1        0
2        0
3        0
4        0
        ..
45203    1
45205    1
45206    1
45209    0
```

```
45210    0
Name: y, Length: 37303, dtype: object
```

[78]:
```
rf.fit(X,y.astype(int))
```

[78]:
```
RandomForestClassifier(random_state=1)
```

[79]:
```
I = rf.feature_importances_
```

[80]:
```
I
```

[80]:
```
array([0.16843324, 0.07493019, 0.02846266, 0.0357738 , 0.00286004,
       0.21485109, 0.02604091, 0.01490493, 0.38394217, 0.04980096])
```

[81]:
```
X.columns
```

[81]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'balance', 'housing',
       'loan', 'duration', 'campaign'],
      dtype='object')
```

[82]:
```
# Define a variable that contains only the feature column names
name_features = X.columns
```

[83]:
```
name_features
```

[83]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'balance', 'housing',
       'loan', 'duration', 'campaign'],
      dtype='object')
```

[84]:
```
len(name_features)
```

[84]:
```
10
```

[85]:
```
len(I)
```

[85]:
```
10
```

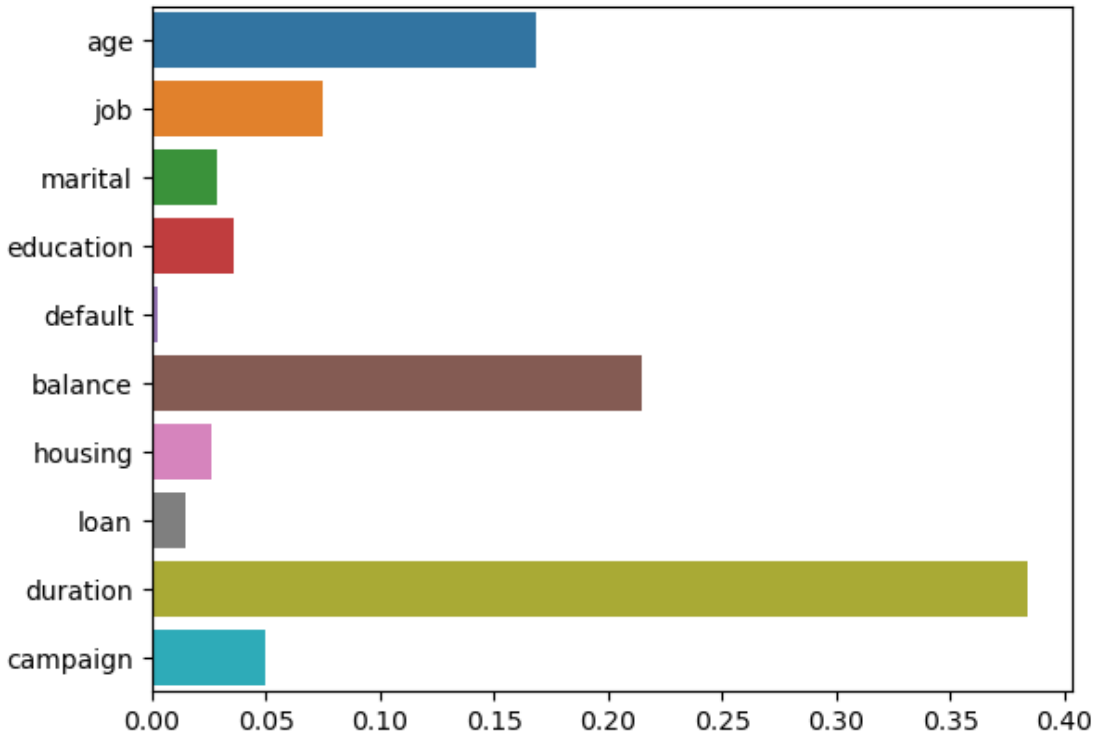
[86]:
```
# generate a new dataframe call it df_I

df_I=pd.DataFrame([I, name_features])
```

[87]:
```
df_I
```

[87]:
```
          0        1         2          3        4         5         6  \
0  0.168433  0.07493  0.028463   0.035774  0.00286  0.214851  0.026041
1       age      job   marital  education  default   balance   housing

          7         8         9
```

```
0  0.014905  0.383942  0.049801
1      loan  duration  campaign
```

[88]:
```
sns.barplot(data=df_I, y=name_features, x=I);
```



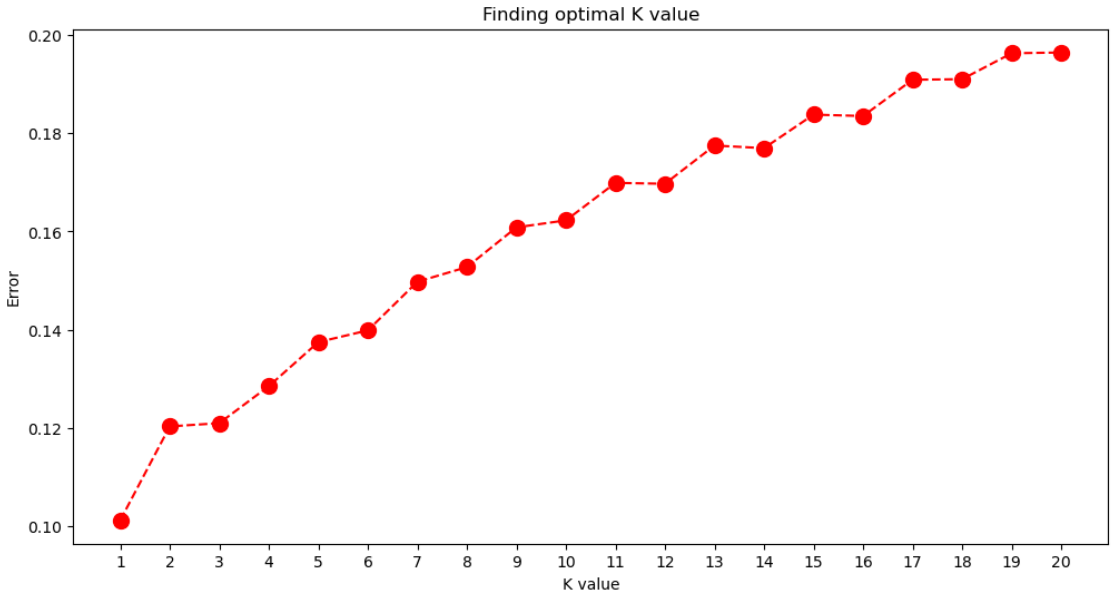
1.0.2 optimal value for KNN

[89]:
```
# find the best value of K in KNN

from sklearn.metrics import balanced_accuracy_score
```

[90]:
```
error=[]

#calculating the error for k values between 1 and 20

for i in range(1,21):  # i=1,2,...,20
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    accuracy = balanced_accuracy_score(y_test, pred_i)
    error.append(1-accuracy)
```

```
error;
```

[91]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(12,6))

plt.plot(range(1,21), error, color='red',   linestyle='dashed', marker='o',␣
 ↪markerfacecolor='red', markersize=10)

plt.title ('Finding optimal K value')
plt.xlabel ('K value')
plt.ylabel('Error');

# Set the ticks on the X-axis to integer values
plt.xticks(ticks=range(1, 21));  # Set the ticks to integers 1 to 20
```



[ ]:
```
######################   Conditional Probability #################
```

[4]:
```
df_cond = pd.read_csv('bank-full.csv',delimiter=';')
```

[5]:
```
df_cond
```

[5]:

	age	job	marital	education	default	balance	housing	loan	\
0	58.0	management	married	tertiary	no	2143	yes	no	
1	44.0	technician	single	NaN	no	29	yes	no	
2	33.0	entrepreneur	married	secondary	no	2	yes	yes	
3	47.0	blue-collar	married	unknown	no	1506	yes	no	

```
4      33.0       unknown    single    unknown      no        1      no   no
…       …           …        …          …          …       …      …   …
45206  51.0    technician   married   tertiary      no      825      no   no
45207  71.0       retired  divorced    primary      no     1729      no   no
45208  72.0       retired   married  secondary      no     5715      no   no
45209  57.0   blue-collar   married  secondary      no      668      no   no
45210  37.0  entrepreneur   married  secondary      no     2971      no   no

          contact  day month  duration  campaign  pdays  previous poutcome    y
0         unknown    5   may       261         1     -1         0  unknown   no
1         unknown    5   may       151         1     -1         0  unknown   no
2         unknown    5   may        76         1     -1         0  unknown   no
3         unknown    5   may        92         1     -1         0  unknown   no
4         unknown    5   may       198         1     -1         0  unknown   no
…           …  …   …        …        …    …        …       …  …
45206   cellular   17   nov       977         3     -1         0  unknown  yes
45207   cellular   17   nov       456         2     -1         0  unknown  yes
45208   cellular   17   nov      1127         5    184         3  success  yes
45209  telephone   17   nov       508         4     -1         0  unknown   no
45210   cellular   17   nov       361         2    188        11    other   no

[45211 rows x 17 columns]
```

[8]:
```
((df_cond['loan']=='yes') & (df_cond['y']=='yes')).sum()
```

[8]:
```
484
```

[10]:
```
# the probability that a person takes term deposit given he has loan???
prob_loan_yes = (df_cond[df_cond["loan"]=="yes"]["y"] == "yes").sum()/
 ↪(df_cond[df_cond["loan"] == 'yes']["loan"]).count()
prob_loan_yes.round(3)
```

[10]:
```
0.067
```

[68]:
```
# the probability that a person makes a term deposit given he has taken housing␣
 ↪loan ???
prob_loan_yes_housing = (df_cond[df_cond["housing"]=="yes"]["y"] == "yes").
 ↪sum()/(df_cond[df_cond["housing"] == 'yes']["housing"]).count()
prob_loan_yes_housing.round(3)
```

[68]:
```
0.077
```

[69]:
```
# the probability of a person making term deposit given he has taken both␣
 ↪housing loan as well as normal loan???
```

```
prob_both_yes = (df_cond[(df_cond["housing"] == "yes") & (df_cond["loan"] ==␣
 ↪"yes")]["y"] == "yes").sum() / len(df_cond[(df_cond["housing"] == "yes") &␣
 ↪(df_cond["loan"] == "yes")])

prob_both_yes.round(3)
```

[69]:
```
0.061
```

[70]:
```
# the probability that he makes term deposit given his age is less than 40??
prob_loan_yes_age = (df_cond[df_cond["age"]<40]["y"] == "yes").sum()/
 ↪(df_cond[df_cond["age"] < 40]["age"]).count()
prob_loan_yes_age.round(3)
```

[70]:
```
0.122
```

[71]:
```
df_cond
```

[71]:
```
       age           job   marital  education default  balance housing loan  \
0       58    management   married   tertiary      no     2143     yes   no
1       44    technician    single  secondary      no       29     yes   no
2       33  entrepreneur   married  secondary      no        2     yes  yes
3       47   blue-collar   married    unknown      no     1506     yes   no
4       33       unknown    single    unknown      no        1      no   no
…      …             …       …         …      …      …   …
45206   51    technician   married   tertiary      no      825      no   no
45207   71       retired  divorced    primary      no     1729      no   no
45208   72       retired   married  secondary      no     5715      no   no
45209   57   blue-collar   married  secondary      no      668      no   no
45210   37  entrepreneur   married  secondary      no     2971      no   no

         contact  day month  duration  campaign  pdays  previous poutcome    y
0        unknown    5   may       261         1     -1         0  unknown   no
1        unknown    5   may       151         1     -1         0  unknown   no
2        unknown    5   may        76         1     -1         0  unknown   no
3        unknown    5   may        92         1     -1         0  unknown   no
4        unknown    5   may       198         1     -1         0  unknown   no
…          …  …   …        …         …    …        …      …  …
45206   cellular   17   nov       977         3     -1         0  unknown  yes
45207   cellular   17   nov       456         2     -1         0  unknown  yes
45208   cellular   17   nov      1127         5    184         3  success  yes
45209  telephone   17   nov       508         4     -1         0  unknown   no
45210   cellular   17   nov       361         2    188        11    other   no

[45211 rows x 17 columns]
```

[72]:
```
# the probability that a person has a housing loan given that he/she is single
```

```
prob_housing_loan_yes_age = (df_cond[df_cond["marital"]=="single"]["housing"]␣
  ↪== "yes").sum()/(df_cond[df_cond["marital"] == "single"]["marital"]).count()
prob_housing_loan_yes_age.round(3)
```

[72]: 0.54

[15]:
```
# The feature importance of loan
FI_loan=(df_cond[df_cond["loan"]=="yes"]["y"] == "yes").sum()/
  ↪(df_cond["loan"]=="yes").sum() - (df_cond[df_cond["loan"]=="no"]["y"] ==␣
  ↪"yes").sum()/(df_cond["loan"]=="no").sum()
FI_loan
```

[15]: -0.05974335845754743

[16]:
```
# The feature importance of housing
FI_housing=(df_cond[df_cond["housing"]=="yes"]["y"] == "yes").sum()/
  ↪(df_cond["housing"]=="yes").sum() - (df_cond[df_cond["housing"]=="no"]["y"]␣
  ↪== "yes").sum()/(df_cond["housing"]=="no").sum()
FI_housing
```

[16]: -0.09002395253461445

[73]:
```
df_linear=pd.read_csv("House Price India.csv")
```

[74]:
```
df_linear
```

[74]:

	number of bedrooms	number of bathrooms	living area	lot area	\
0	5	2.50	3650	9050	
1	4	2.50	2920	4000	
2	5	2.75	2910	9480	
3	4	2.50	3310	42998	
4	3	2.00	2710	4500	
…	…	…	…	…	
14615	2	1.50	1556	20000	
14616	3	2.00	1680	7000	
14617	2	1.00	1070	6120	
14618	4	1.00	1030	6621	
14619	3	1.00	900	4770	

	number of floors	waterfront present	number of views	\
0	2.0	0	4	
1	1.5	0	0	
2	1.5	0	0	
3	2.0	0	0	
4	1.5	0	0	
…	…	…	…	
14615	1.0	0	0	

```
14616               1.5                    0                0
14617               1.0                    0                0
14618               1.0                    0                0
14619               1.0                    0                0

       condition of the house  grade of the house  \
0                           5                  10
1                           5                   8
2                           3                   8
3                           3                   9
4                           4                   8
…                         …                   …
14615                       4                   7
14616                       4                   7
14617                       3                   6
14618                       4                   6
14619                       3                   6

       Area of the house(excluding basement)  Area of the basement  \
0                                       3370                   280
1                                       1910                  1010
2                                       2910                     0
3                                       3310                     0
4                                       1880                   830
…                                       …                    …
14615                                   1556                     0
14616                                   1680                     0
14617                                   1070                     0
14618                                   1030                     0
14619                                    900                     0

       Built Year  living_area_renov  lot_area_renov  \
0            1921               2880            5400
1            1909               2470            4000
2            1939               2940            6600
3            2001               3350           42847
4            1929               2060            4500
…           …                  …               …
14615        1957               2250           17286
14616        1968               1540            7480
14617        1962               1130            6120
14618        1955               1420            6631
14619        1969                900            3480

       Number of schools nearby  Distance from the airport    Price
0                             2                         58  2380000
1                             2                         51  1400000
```

```
2                              1                          53  1200000
3                              3                          76   838000
4                              1                          51   805000
…                            …                          …      …
14615                          3                          76   221700
14616                          3                          59   219200
14617                          2                          64   209000
14618                          3                          54   205000
14619                          2                          55   146000

[14620 rows x 17 columns]
```

[75]:
```
df_linear.head(5)
```

[75]:
```
   number of bedrooms  number of bathrooms  living area  lot area  \
0                   5                 2.50         3650      9050
1                   4                 2.50         2920      4000
2                   5                 2.75         2910      9480
3                   4                 2.50         3310     42998
4                   3                 2.00         2710      4500

   number of floors  waterfront present  number of views  \
0               2.0                   0                4
1               1.5                   0                0
2               1.5                   0                0
3               2.0                   0                0
4               1.5                   0                0

   condition of the house  grade of the house  \
0                       5                  10
1                       5                   8
2                       3                   8
3                       3                   9
4                       4                   8

   Area of the house(excluding basement)  Area of the basement  Built Year  \
0                                   3370                   280        1921
1                                   1910                  1010        1909
2                                   2910                     0        1939
3                                   3310                     0        2001
4                                   1880                   830        1929

   living_area_renov  lot_area_renov  Number of schools nearby  \
0               2880            5400                         2
1               2470            4000                         2
2               2940            6600                         1
3               3350           42847                         3
```

```
4                2060            4500                          1

   Distance from the airport    Price
0                         58  2380000
1                         51  1400000
2                         53  1200000
3                         76   838000
4                         51   805000
```

[76]:
```
df_linear.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14620 entries, 0 to 14619
Data columns (total 17 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   number of bedrooms                      14620 non-null  int64
 1   number of bathrooms                     14620 non-null  float64
 2   living area                             14620 non-null  int64
 3   lot area                                14620 non-null  int64
 4   number of floors                        14620 non-null  float64
 5   waterfront present                      14620 non-null  int64
 6   number of views                         14620 non-null  int64
 7   condition of the house                  14620 non-null  int64
 8   grade of the house                      14620 non-null  int64
 9   Area of the house(excluding basement)   14620 non-null  int64
 10  Area of the basement                    14620 non-null  int64
 11  Built Year                              14620 non-null  int64
 12  living_area_renov                       14620 non-null  int64
 13  lot_area_renov                          14620 non-null  int64
 14  Number of schools nearby                14620 non-null  int64
 15  Distance from the airport               14620 non-null  int64
 16  Price                                   14620 non-null  int64
dtypes: float64(2), int64(15)
memory usage: 1.9 MB
```

[77]:
```
X=df_linear.iloc[:,0:14]
y=df_linear["Price"]
X1=df_linear.iloc[:,0].values
```

[80]:
```
lin_reg=LinearRegression()
```

[81]:
```
lin_reg
```

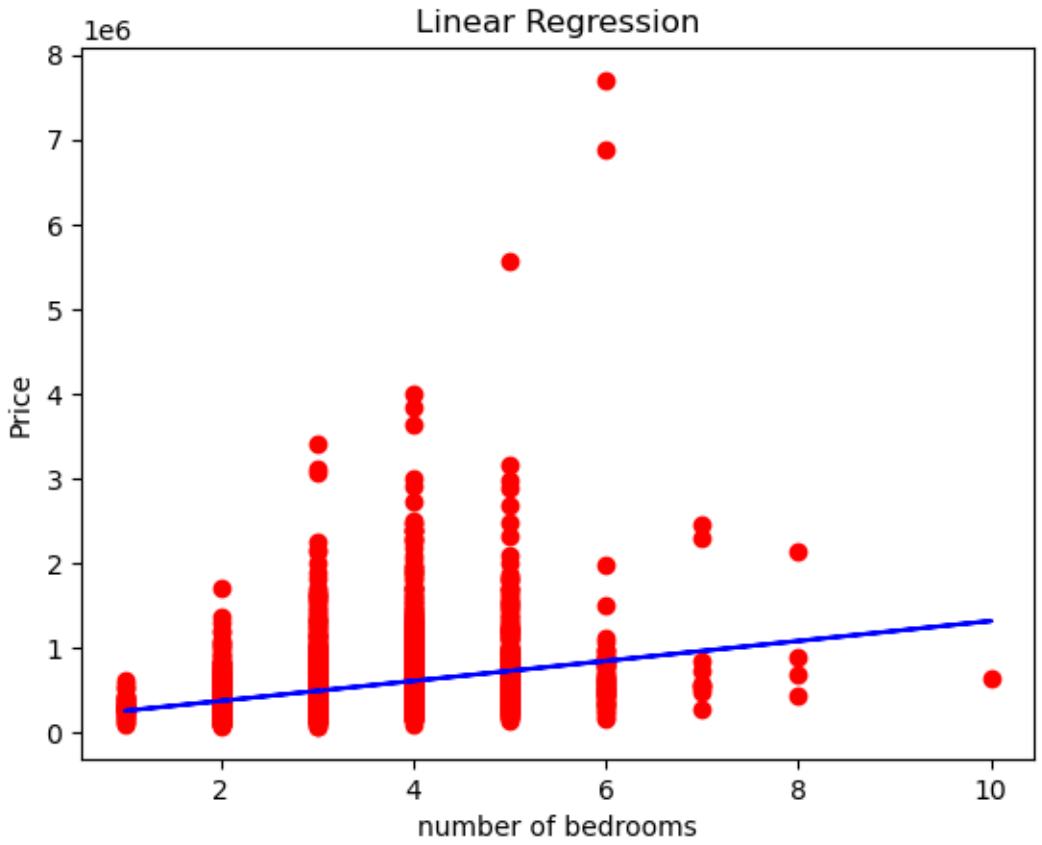
[81]:
```
LinearRegression()
```

[82]:
```
Xtrain, Xtest, ytrain, ytest = train_test_split (X1, y, test_size=0.3,␣
↪random_state=0)
```

```
lin_reg.fit(Xtrain.reshape(-1,1), ytrain)
```

[82]:
```
LinearRegression()
```

[84]:
```
ypred=lin_reg.predict(Xtest.reshape(-1,1))
#correlation between number of bedrooms and price
plt.scatter(Xtest, ytest, color='red')
plt.plot(Xtest, ypred, color='blue')
plt.title('Linear Regression')
plt.xlabel('number of bedrooms')
plt.ylabel('Price');
```



[87]:
```
X=df_linear.iloc[:,0:14]
y=df_linear["Price"]
X3=df_linear.iloc[:,2].values
type(X3)
X3;
```

[88]:
```
lin_reg=LinearRegression()
lin_reg
```

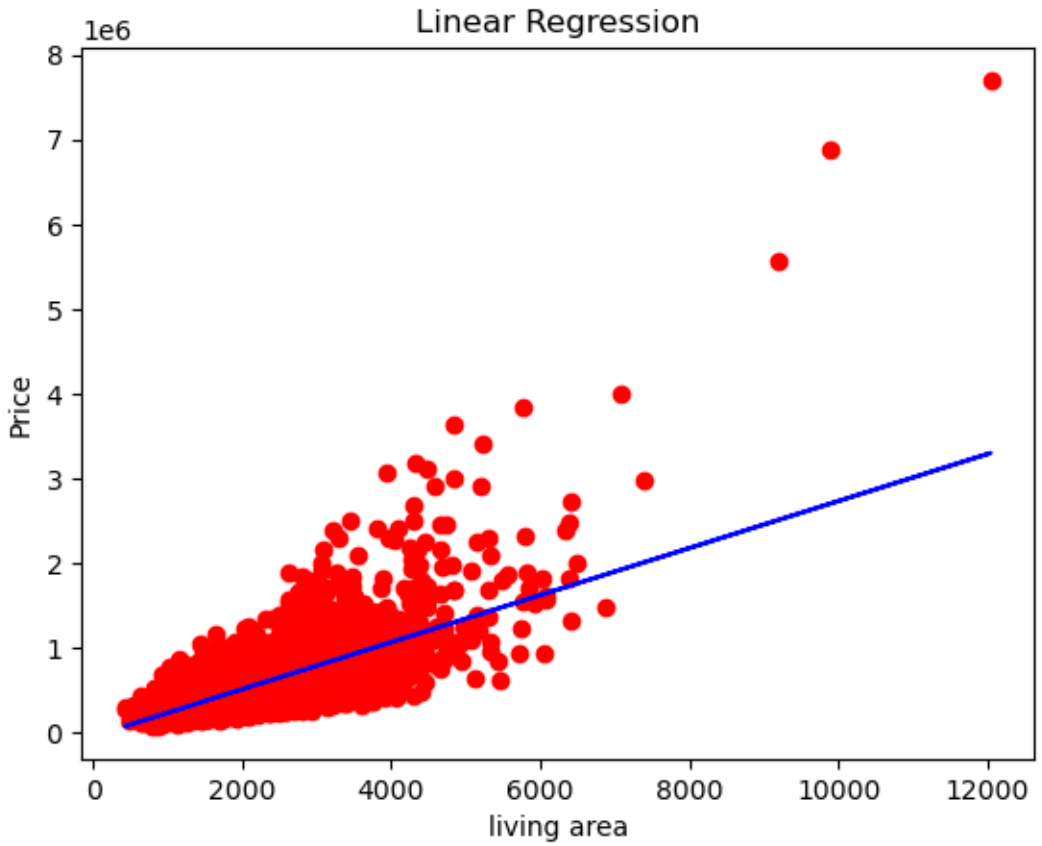
[88]:
```
LinearRegression()
```

[89]:
```
Xtrain, Xtest, ytrain, ytest = train_test_split (X3, y, test_size=0.3,␣
↪random_state=0)
lin_reg.fit(Xtrain.reshape(-1,1), ytrain)
```

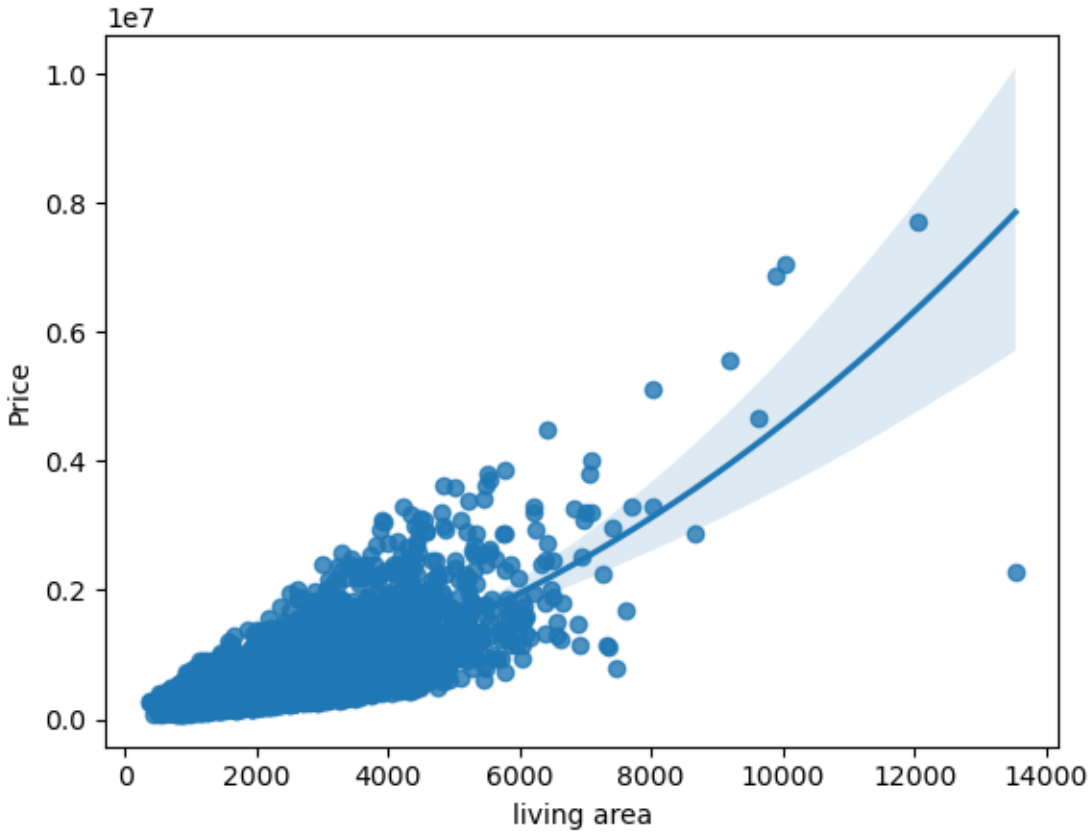
[89]:
```
LinearRegression()
```

[90]:
```
ypred=lin_reg.predict(Xtest.reshape(-1,1))
```

[91]:
```
plt.scatter(Xtest, ytest, color='red')
plt.plot(Xtest, ypred, color='blue')
plt.title('Linear Regression')
plt.xlabel('living area')
plt.ylabel('Price');
```



[92]:
```
sns.regplot(x="living area", y="Price", data=df_linear, order=2);
```



[94]:
```
#correlation between grade of the house and price
X=df_linear.iloc[:,0:14]
y=df_linear["Price"]
X7=df_linear.iloc[:,6].values
type(X7)
X7;
```

[98]:
```
lin_reg=LinearRegression()
lin_reg
```

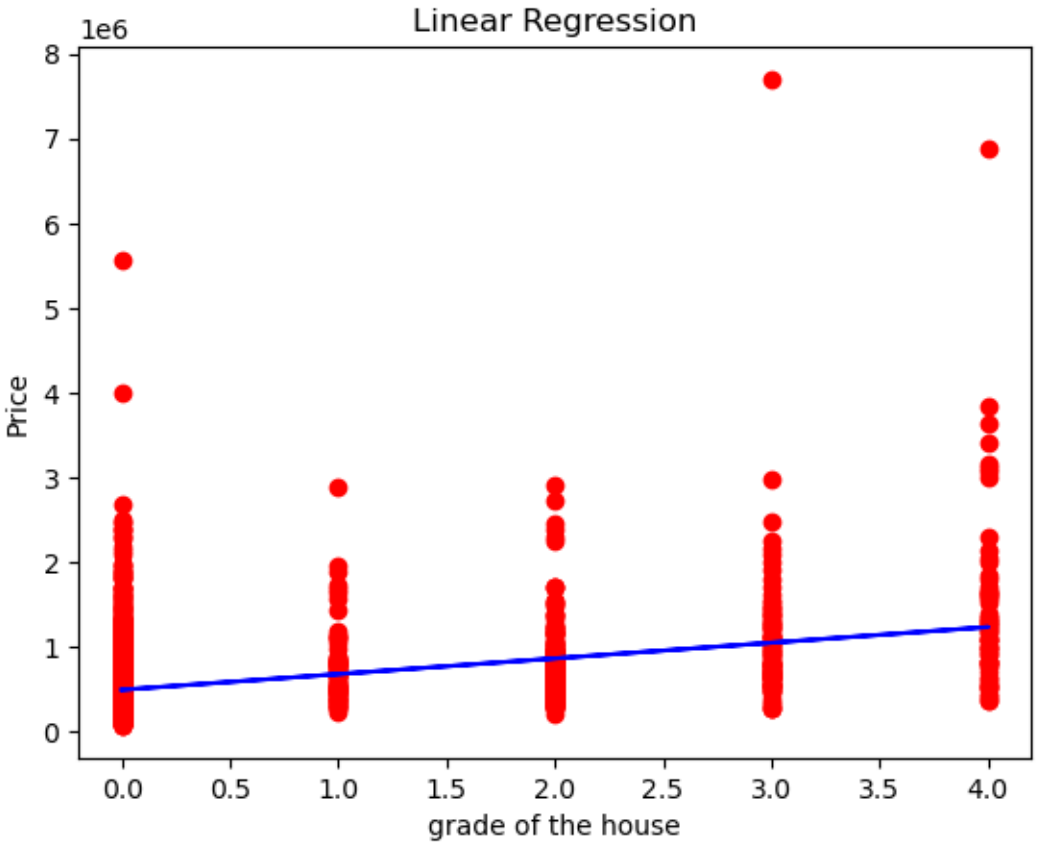
[98]:
```
LinearRegression()
```

[99]:
```
Xtrain, Xtest, ytrain, ytest = train_test_split (X7, y, test_size=0.3,␣
↪random_state=0)
lin_reg.fit(Xtrain.reshape(-1,1), ytrain)
```

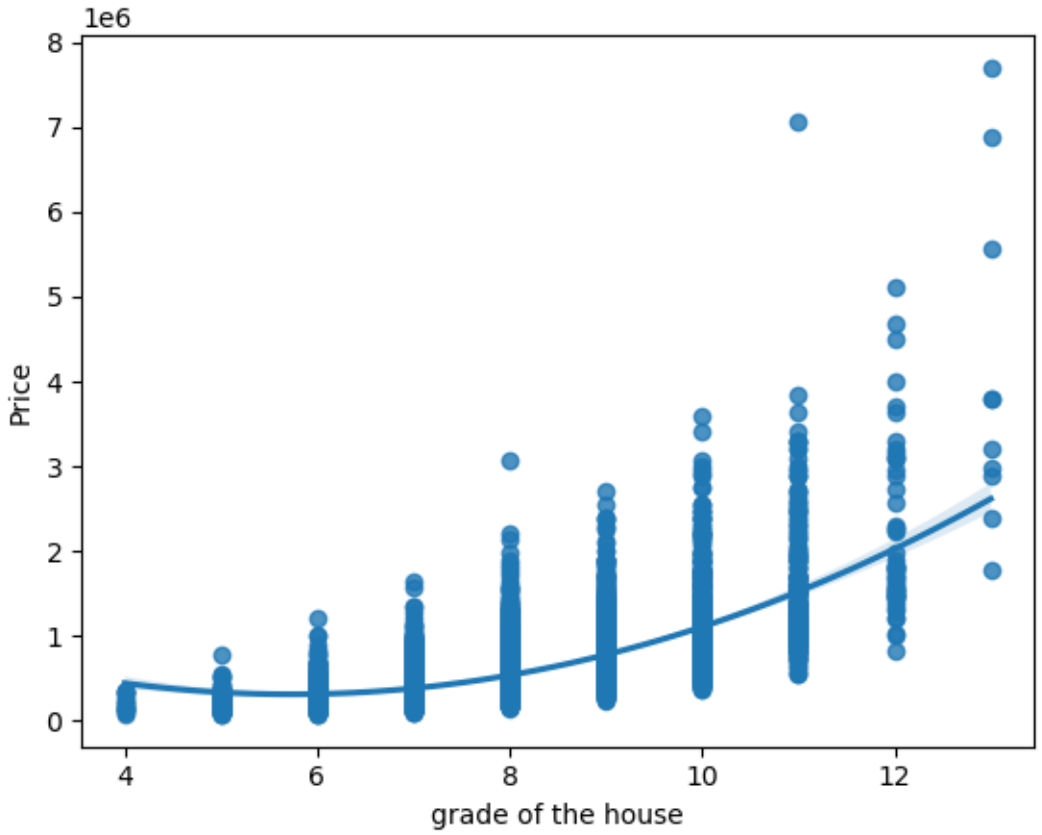
[99]:
```
LinearRegression()
```

[100]:
```
ypred=lin_reg.predict(Xtest.reshape(-1,1))
```

[101]:
```
plt.scatter(Xtest, ytest, color='red')
plt.plot(Xtest, ypred, color='blue')
plt.title('Linear Regression')
plt.xlabel('grade of the house')
plt.ylabel('Price');
```



[102]:
```
sns.regplot(x="grade of the house", y="Price", data=df_linear, order=2);
```



[103]:
```
df=pd.read_csv("House Price India.csv")
```

[104]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14620 entries, 0 to 14619
Data columns (total 17 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   number of bedrooms                     14620 non-null  int64
 1   number of bathrooms                    14620 non-null  float64
 2   living area                            14620 non-null  int64
 3   lot area                               14620 non-null  int64
 4   number of floors                       14620 non-null  float64
 5   waterfront present                     14620 non-null  int64
 6   number of views                        14620 non-null  int64
 7   condition of the house                 14620 non-null  int64
 8   grade of the house                     14620 non-null  int64
 9   Area of the house(excluding basement)  14620 non-null  int64
 10  Area of the basement                   14620 non-null  int64
 11  Built Year                             14620 non-null  int64
```

```
 12  living_area_renov                    14620 non-null  int64
 13  lot_area_renov                       14620 non-null  int64
 14  Number of schools nearby             14620 non-null  int64
 15  Distance from the airport            14620 non-null  int64
 16  Price                                14620 non-null  int64
dtypes: float64(2), int64(15)
memory usage: 1.9 MB
```

[105]:
```
df.head(5)
```

[105]:

	number of bedrooms	number of bathrooms	living area	lot area
0	5	2.50	3650	9050
1	4	2.50	2920	4000
2	5	2.75	2910	9480
3	4	2.50	3310	42998
4	3	2.00	2710	4500

	number of floors	waterfront present	number of views
0	2.0	0	4
1	1.5	0	0
2	1.5	0	0
3	2.0	0	0
4	1.5	0	0

	condition of the house	grade of the house
0	5	10
1	5	8
2	3	8
3	3	9
4	4	8

	Area of the house(excluding basement)	Area of the basement	Built Year
0	3370	280	1921
1	1910	1010	1909
2	2910	0	1939
3	3310	0	2001
4	1880	830	1929

	living_area_renov	lot_area_renov	Number of schools nearby
0	2880	5400	2
1	2470	4000	2
2	2940	6600	1
3	3350	42847	3
4	2060	4500	1

	Distance from the airport	Price
0	58	2380000

```
1                          51  1400000
2                          53  1200000
3                          76   838000
4                          51   805000
```

[106]:
```
# Calculate the correlation between input variables, as well as between inputs and output variables.
# Calculate the correlation matrix for the dataset
correlation_matrix = df.corr()
correlation_matrix
```

[106]:
```
                                        number of bedrooms  \
number of bedrooms                                1.000000
number of bathrooms                               0.509784
living area                                       0.570526
lot area                                          0.034416
number of floors                                  0.177294
waterfront present                               -0.006257
number of views                                   0.078665
condition of the house                            0.026597
grade of the house                                0.352945
Area of the house(excluding basement)             0.473599
Area of the basement                              0.300332
Built Year                                        0.152954
living_area_renov                                 0.389855
lot_area_renov                                    0.029400
Number of schools nearby                          0.003397
Distance from the airport                        -0.006157
Price                                             0.308460

                                        number of bathrooms  living area  \
number of bedrooms                                 0.509784     0.570526
number of bathrooms                                1.000000     0.753517
living area                                        0.753517     1.000000
lot area                                           0.080806     0.174420
number of floors                                   0.502924     0.354743
waterfront present                                 0.060104     0.105837
number of views                                    0.183789     0.287728
condition of the house                            -0.128232    -0.063358
grade of the house                                 0.663054     0.761835
Area of the house(excluding basement)              0.684391     0.875793
Area of the basement                               0.287190     0.441491
Built Year                                         0.498127     0.309602
living_area_renov                                  0.570530     0.757571
lot_area_renov                                     0.078627     0.180312
Number of schools nearby                           0.002180     0.002370
Distance from the airport                          0.009206     0.002511
```

```
Price                                          0.531735     0.712169

                                    lot area  number of floors  \
number of bedrooms                  0.034416          0.177294
number of bathrooms                 0.080806          0.502924
living area                         0.174420          0.354743
lot area                            1.000000         -0.004138
number of floors                   -0.004138          1.000000
waterfront present                  0.026282          0.016316
number of views                     0.078308          0.020153
condition of the house             -0.008548         -0.269928
grade of the house                  0.110546          0.463082
Area of the house(excluding basement)  0.183553          0.525643
Area of the basement                0.019755         -0.242976
Built Year                          0.051615          0.481565
living_area_renov                   0.149744          0.285093
lot_area_renov                      0.706812         -0.010120
Number of schools nearby           -0.012671         -0.007579
Distance from the airport           0.003291          0.016567
Price                               0.081992          0.262732

                                    waterfront present  number of views  \
number of bedrooms                           -0.006257         0.078665
number of bathrooms                           0.060104         0.183789
living area                                   0.105837         0.287728
lot area                                      0.026282         0.078308
number of floors                              0.016316         0.020153
waterfront present                            1.000000         0.400206
number of views                               0.400206         1.000000
condition of the house                        0.018644         0.052533
grade of the house                            0.079831         0.254532
Area of the house(excluding basement)         0.071865         0.162672
Area of the basement                          0.085441         0.293062
Built Year                                   -0.024226        -0.055357
living_area_renov                             0.085743         0.281452
lot_area_renov                                0.032055         0.072300
Number of schools nearby                      0.001563         0.008004
Distance from the airport                     0.001448        -0.001657
Price                                         0.263687         0.395973

                                    condition of the house  \
number of bedrooms                                0.026597
number of bathrooms                              -0.128232
living area                                      -0.063358
lot area                                         -0.008548
number of floors                                 -0.269928
waterfront present                                0.018644
```

number of views	0.052533
condition of the house	1.000000
grade of the house	-0.152530
Area of the house(excluding basement)	-0.167695
Area of the basement	0.180609
Built Year	-0.381718
living_area_renov	-0.099743
lot_area_renov	-0.004748
Number of schools nearby	-0.006939
Distance from the airport	-0.002136
Price	0.041376

	grade of the house \
number of bedrooms	0.352945
number of bathrooms	0.663054
living area	0.761835
lot area	0.110546
number of floors	0.463082
waterfront present	0.079831
number of views	0.254532
condition of the house	-0.152530
grade of the house	1.000000
Area of the house(excluding basement)	0.758222
Area of the basement	0.167160
Built Year	0.440358
living_area_renov	0.720019
lot_area_renov	0.116725
Number of schools nearby	0.000986
Distance from the airport	0.004940
Price	0.671814

	Area of the house(excluding basement) \
number of bedrooms	0.473599
number of bathrooms	0.684391
living area	0.875793
lot area	0.183553
number of floors	0.525643
waterfront present	0.071865
number of views	0.162672
condition of the house	-0.167695
grade of the house	0.758222
Area of the house(excluding basement)	1.000000
Area of the basement	-0.046445
Built Year	0.419369
living_area_renov	0.737744
lot_area_renov	0.194670
Number of schools nearby	-0.002894

Distance from the airport	0.001222
Price	0.615220

	Area of the basement	Built Year
number of bedrooms	0.300332	0.152954
number of bathrooms	0.287190	0.498127
living area	0.441491	0.309602
lot area	0.019755	0.051615
number of floors	-0.242976	0.481565
waterfront present	0.085441	-0.024226
number of views	0.293062	-0.055357
condition of the house	0.180609	-0.381718
grade of the house	0.167160	0.440358
Area of the house(excluding basement)	-0.046445	0.419369
Area of the basement	1.000000	-0.138843
Built Year	-0.138843	1.000000
living_area_renov	0.196403	0.328625
lot_area_renov	0.011283	0.072874
Number of schools nearby	0.010284	-0.001631
Distance from the airport	0.002926	-0.003968
Price	0.330202	0.050307

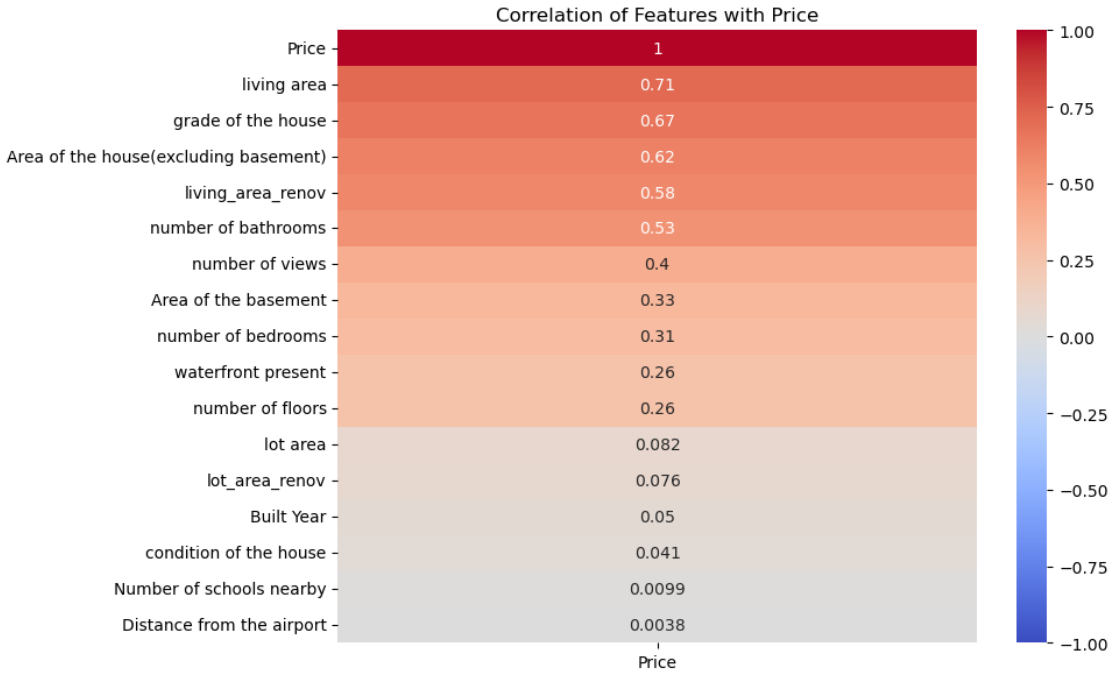
	living_area_renov	lot_area_renov
number of bedrooms	0.389855	0.029400
number of bathrooms	0.570530	0.078627
living area	0.757571	0.180312
lot area	0.149744	0.706812
number of floors	0.285093	-0.010120
waterfront present	0.085743	0.032055
number of views	0.281452	0.072300
condition of the house	-0.099743	-0.004748
grade of the house	0.720019	0.116725
Area of the house(excluding basement)	0.737744	0.194670
Area of the basement	0.196403	0.011283
Built Year	0.328625	0.072874
living_area_renov	1.000000	0.189225
lot_area_renov	0.189225	1.000000
Number of schools nearby	-0.001203	-0.025014
Distance from the airport	-0.005673	-0.014587
Price	0.584924	0.075535

	Number of schools nearby
number of bedrooms	0.003397
number of bathrooms	0.002180
living area	0.002370
lot area	-0.012671
number of floors	-0.007579

```
waterfront present                                0.001563
number of views                                   0.008004
condition of the house                           -0.006939
grade of the house                                0.000986
Area of the house(excluding basement)            -0.002894
Area of the basement                              0.010284
Built Year                                       -0.001631
living_area_renov                                -0.001203
lot_area_renov                                   -0.025014
Number of schools nearby                          1.000000
Distance from the airport                         0.004035
Price                                             0.009890
```

	Distance from the airport	Price
number of bedrooms	-0.006157	0.308460
number of bathrooms	0.009206	0.531735
living area	0.002511	0.712169
lot area	0.003291	0.081992
number of floors	0.016567	0.262732
waterfront present	0.001448	0.263687
number of views	-0.001657	0.395973
condition of the house	-0.002136	0.041376
grade of the house	0.004940	0.671814
Area of the house(excluding basement)	0.001222	0.615220
Area of the basement	0.002926	0.330202
Built Year	-0.003968	0.050307
living_area_renov	-0.005673	0.584924
lot_area_renov	-0.014587	0.075535
Number of schools nearby	0.004035	0.009890
Distance from the airport	1.000000	0.003804
Price	0.003804	1.000000

[107]:
```
plt.figure(figsize=(10, 6))  # Adjust width and height as needed
sns.heatmap(correlation_matrix[['Price']].sort_values(by='Price',␣
 ↪ascending=False),
            annot=True, cmap='coolwarm', vmin=-1, vmax=1, cbar=True)
plt.yticks(rotation=0)

plt.title('Correlation of Features with Price')
plt.tight_layout()  # Adjusts the plot to fit within the figure area
plt.show()
```



[108]:
```
#Living area has the highest positive correlation with price (0.71), indicating␣
 ↪a strong relationship.
#Grade of the house also has a significant positive correlation with price (0.
 ↪67).
#Area of the house (excluding basement) and living_area_renov show moderate␣
 ↪positive correlations with price (0.61 and 0.58, respectively).
#Features like number of bathrooms (0.53), area of the basement (0.33), and␣
 ↪number of bedrooms (0.31) have lower but still relevant positive␣
 ↪correlations with price.
#Features like lot area, built year, condition of the house, and distance from␣
 ↪the airport have minimal correlation with price.
```

[109]:
```
X = df.drop(columns=["Price"])
y = df['Price']
```

[110]:
```
# Splitting the dataset into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=0)
# Initialize and train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)
```

[110]:
```
LinearRegression()
```

[111]:
```
# Make predictions on the test set
y_pred = model.predict(X_test)
coefficients = pd.DataFrame(model.coef_, X.columns, columns=['Coefficient'])
intercept = model.intercept_
```

[112]:
```
mae = mean_absolute_error(y_test, y_pred)
```

[115]:
```
X = sm.add_constant(X)
reg = sm.OLS(y, X).fit()
print(reg.summary())
# output
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.664
Model:                            OLS   Adj. R-squared:                  0.664
Method:                 Least Squares   F-statistic:                     1926.
Date:                Fri, 04 Oct 2024   Prob (F-statistic):               0.00
Time:                        23:46:26   Log-Likelihood:            -2.0012e+05
No. Observations:               14620   AIC:                         4.003e+05
Df Residuals:                   14604   BIC:                         4.004e+05
Df Model:                          15
Covariance Type:            nonrobust
==============================================================================
                                                 coef    std err          t
P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const                                     6.156e+06   1.57e+05     39.153
0.000    5.85e+06    6.46e+06
number of bedrooms                       -3.984e+04   2387.221    -16.688
0.000   -4.45e+04   -3.52e+04
number of bathrooms                       4.652e+04   4141.671     11.232
0.000    3.84e+04    5.46e+04
living area                                114.4892      2.876     39.814
0.000     108.853     120.126
lot area                                    -0.1311      0.066     -1.985
0.047      -0.261      -0.002
number of floors                          2.474e+04   4536.891      5.454
0.000    1.59e+04    3.36e+04
waterfront present                         5.77e+05   2.21e+04     26.078
0.000    5.34e+05     6.2e+05
number of views                           4.217e+04   2736.267     15.410
0.000    3.68e+04    4.75e+04
condition of the house                    2.151e+04   2921.274      7.364
0.000    1.58e+04    2.72e+04
grade of the house                        1.189e+05   2712.828     43.834
```

```
0.000    1.14e+05    1.24e+05
Area of the house(excluding basement)    58.5015      2.827     20.691
0.000      52.959      64.043
Area of the basement                     55.9878      3.313     16.901
0.000      49.495      62.481
Built Year                            -3550.9701     80.317    -44.212
0.000   -3708.402   -3393.538
living_area_renov                        13.0214      4.328      3.009
0.003       4.538      21.504
lot_area_renov                           -0.5166      0.097     -5.331
0.000      -0.707      -0.327
Number of schools nearby                3040.0393   2157.430      1.409
0.159   -1188.796    7268.875
Distance from the airport               -113.8137    197.351     -0.577
0.564    -500.647     273.019
==============================================================================
Omnibus:                    11983.201   Durbin-Watson:                   1.624
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1201714.919
Skew:                           3.347   Prob(JB):                         0.00
Kurtosis:                      46.908   Cond. No.                     2.95e+17
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The smallest eigenvalue is 3.76e-22. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

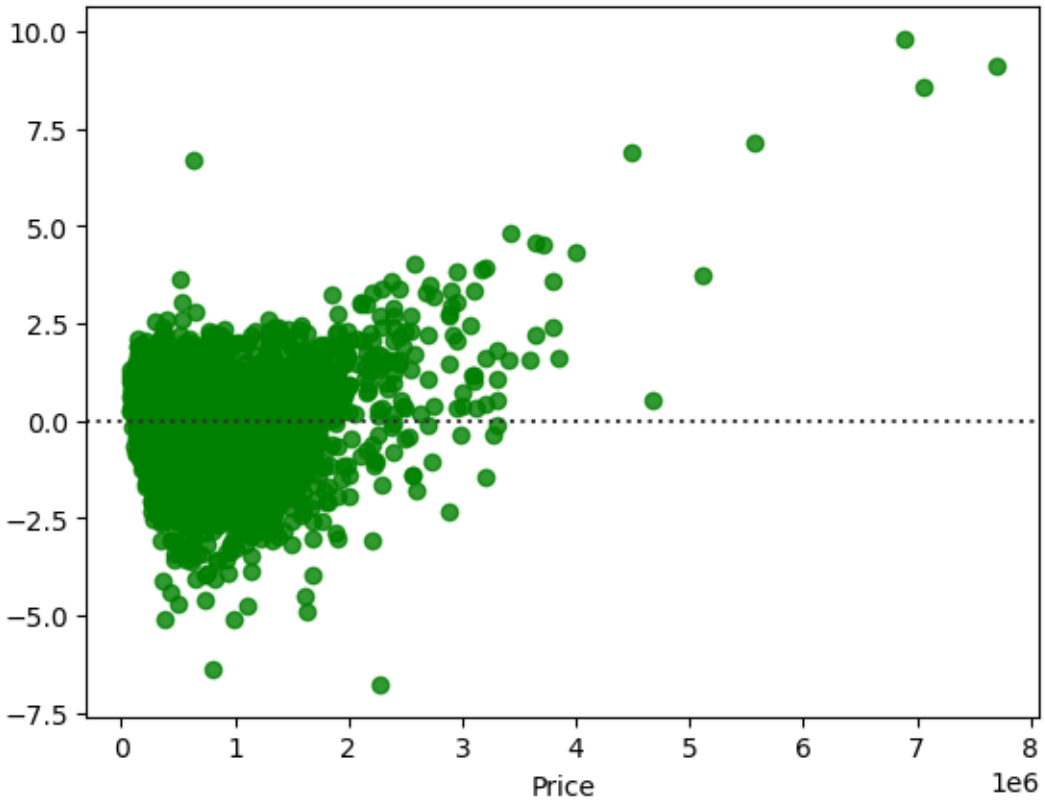
[116]:
```
df['estimated']=reg.predict(X)
y=df['Price']
yhat=df['estimated']

residual = y-yhat

df['residual']=residual
```

[119]:
```
res= stat()
res.reg_metric(y=np.array(y), yhat=np.array(reg.predict(X)), resid=np.array(reg.
 ↪resid))
res.reg_metric_df
```

[119]:

	Metrics	Value
0	Root Mean Square Error (RMSE)	2.129635e+05
1	Mean Squared Error (MSE)	4.535346e+10
2	Mean Absolute Error (MAE)	1.366337e+05
3	Mean Absolute Percentage Error (MAPE)	2.851000e-01

[120]:
```
influence=reg.get_influence()
residual_price=influence.resid_studentized_external
sns.residplot(x = 'Price',y = residual_price,data = df,color='g');
```



1.0.3 Polynomial Regression

[123]:
```
poly = PolynomialFeatures(degree=2)
X_poly_train = poly.fit_transform(X_train)
X_poly_test = poly.transform(X_test)
```

[124]:
```
poly_model = LinearRegression()
poly_model.fit(X_poly_train, y_train)
```

[124]:
```
LinearRegression()
```

[125]:
```
y_pred_poly = poly_model.predict(X_poly_test)
```

[126]:
```
Ypred_poly = poly_model.predict(X_poly_train)
residual_poly = np.array(y_train-Ypred_poly)
```

[127]:
```
res= stat()
res.reg_metric(np.array(y_train), np.array(Ypred_poly) , resid=residual_poly)
res.reg_metric_df
```

[127]:

	Metrics	Value
0	Root Mean Square Error (RMSE)	1.773649e+05
1	Mean Squared Error (MSE)	3.145829e+10
2	Mean Absolute Error (MAE)	1.193315e+05
3	Mean Absolute Percentage Error (MAPE)	2.510000e-01

[128]:
```
reg = sm.OLS(y_train, X_poly_train).fit()
reg.summary()
```

[128]:

Dep. Variable:	Price	**R-squared:**	0.765
Model:	OLS	**Adj. R-squared:**	0.763
Method:	Least Squares	**F-statistic:**	283.7
Date:	Fri, 04 Oct 2024	**Prob (F-statistic):**	0.00
Time:	23:54:24	**Log-Likelihood:**	-1.5795e+05
No. Observations:	11696	**AIC:**	3.162e+05
Df Residuals:	11562	**BIC:**	3.172e+05
Df Model:	133		
Covariance Type:	nonrobust		

	coef	std err	t	P> |t|	[0.025	0.975]
const	6.587e+07	1.12e+07	5.892	0.000	4.4e+07	8.78e+07
x1	7.864e+05	1.9e+05	4.141	0.000	4.14e+05	1.16e+06
x2	-1.07e+06	3.31e+05	-3.235	0.001	-1.72e+06	-4.22e+05
x3	2766.1871	845.359	3.272	0.001	1109.141	4423.234
x4	-17.0665	7.663	-2.227	0.026	-32.087	-2.047
x5	-2.047e+06	4.43e+05	-4.620	0.000	-2.92e+06	-1.18e+06
x6	-3.806e+06	1.06e+06	-3.574	0.000	-5.89e+06	-1.72e+06
x7	2.246e+05	2.23e+05	1.006	0.315	-2.13e+05	6.62e+05
x8	3.111e+05	2.51e+05	1.241	0.215	-1.8e+05	8.02e+05
x9	8.794e+05	2.22e+05	3.969	0.000	4.45e+05	1.31e+06
x10	-2410.6967	827.769	-2.912	0.004	-4033.263	-788.130
x11	-3485.0679	859.853	-4.053	0.000	-5170.525	-1799.611
x12	-7.023e+04	1.13e+04	-6.215	0.000	-9.24e+04	-4.81e+04
x13	3921.0954	356.171	11.009	0.000	3222.940	4619.251
x14	-49.7338	11.427	-4.352	0.000	-72.133	-27.335
x15	-3.435e+04	1.83e+05	-0.188	0.851	-3.92e+05	3.24e+05
x16	-1.17e+04	1.67e+04	-0.701	0.483	-4.44e+04	2.1e+04
x17	1364.5574	331.531	4.116	0.000	714.701	2014.414
x18	2632.8968	4537.348	0.580	0.562	-6261.072	1.15e+04
x19	-15.8761	3.199	-4.963	0.000	-22.147	-9.605
x20	0.1575	0.091	1.726	0.084	-0.021	0.336
x21	1.45e+04	6456.183	2.246	0.025	1843.047	2.72e+04
x22	-899.5737	2.72e+04	-0.033	0.974	-5.42e+04	5.24e+04
x23	-2490.1847	3355.248	-0.742	0.458	-9067.039	4086.670
x24	-2996.9839	3773.366	-0.794	0.427	-1.04e+04	4399.452
x25	-4022.7120	3434.962	-1.171	0.242	-1.08e+04	2710.395
x26	-1.5664	3.661	-0.428	0.669	-8.743	5.610
x27	-9.2302	4.025	-2.293	0.022	-17.121	-1.340
x28	-428.9536	98.240	-4.366	0.000	-621.521	-236.386
x29	27.2761	5.497	4.962	0.000	16.501	38.051
x30	-0.0486	0.145	-0.336	0.737	-0.332	0.235
x31	4431.9250	2936.635	1.509	0.131	-1324.377	1.02e+04
x32	383.8766	269.496	1.424	0.154	-144.380	912.134
x33	-9059.6331	5173.316	-1.751	0.080	-1.92e+04	1080.942
x34	20.8859	5.242	3.984	0.000	10.611	31.161
x35	-0.3223	0.161	-2.003	0.045	-0.638	-0.007
x36	-2.975e+04	9501.624	-3.131	0.002	-4.84e+04	-1.11e+04
x37	-6.298e+04	3.86e+04	-1.634	0.102	-1.39e+05	1.26e+04
x38	1.173e+04	5042.938	2.326	0.020	1846.790	2.16e+04
x39	-1439.6685	6367.366	-0.226	0.821	-1.39e+04	1.1e+04
x40	2.018e+04	5731.869	3.521	0.000	8948.092	3.14e+04
x41	20.7011	5.315	3.895	0.000	10.283	31.120
x42	0.5203	6.149	0.085	0.933	-11.533	12.574
x43	497.0107	166.247	2.990	0.003	171.139	822.882
x44	-29.4154	8.471	-3.472	0.001	-46.021	-12.810
x45	0.2802	0.250	1.121	0.262	-0.210	0.770
x46	254.8610	4790.816	0.053	0.958	-9135.948	9645.670
x47	374.5480	436.920	0.857	0.391	-481.889	1230.985
x48	-57.9071	16.045	-3.609	0.000	-89.357	-26.457
x49	-190.0649	52.656	-3.610	0.000	-293.280	-86.850
x50	1.8153	6.928	0.262	0.793	-11.764	15.394
x51	174.6081	18.263	9.561	0.000	138.809	210.407

Omnibus:	4244.626	**Durbin-Watson:**	2.011
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	81692.861
Skew:	1.256	**Prob(JB):**	0.00
Kurtosis:	15.701	**Cond. No.**	1.07e+16

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.07e+16. This might indicate that there are
strong multicollinearity or other numerical problems.

[ ]: