Projects

By Priyanshu Das | SIC : 20BCSE21 | Branch : CSE

Project 1:

a) Prepare a preprocessing dataset.
b) Build a multilinear regression model using data obtained in (a) and also to find the linear coefficient intercept of the Model.
c) Using the model determined in (b). Find the net turnover for the following row vector.
CI = 50661
ES = 115641
AE = 92496
City = Bengaluru

In [1]:
```
'''
Project 1:
a)      Prepare a preprocessing dataset.
b)      Build a multilinear regression model using data obtained in (a) and also to find the linear coefficient intercept of the Model.
c)      Using the model determined in (b). Find the net turnover for the following row vector.
        CI = 50661
        ES = 115641
        AE = 92496
        City = Bengaluru

'''

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder , LabelEncoder,StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split


# Import data set
dataset=pd.read_csv('../Data/investment_data.csv')

# To create feature matrix and dependent variable vector
```

```
x=dataset.iloc[:,:-1].values
y=dataset.iloc[:,-1].values

# Replace missing data
imputer=SimpleImputer(missing_values=np.nan,strategy='mean')
imputer.fit(x[:,:3])
x[:,:3]=imputer.transform(x[:,:3])


# Encoding
# Feature matrix using  OneHotEncoding
ct=ColumnTransformer(transformers=[('encoder',OneHotEncoder(),[3])],remainder='passthrough')
x=np.array(ct.fit_transform(x))

# Splitting of data into training data set and testing data set
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=1)

# Build a multiple linear model
regn=LinearRegression()
regn.fit(xtrain,ytrain)
yestimated=regn.predict(xtest)
np.concatenate((yestimated.reshape(len(yestimated),1),yestimated.reshape(len(yestimated),1)),1)


# Finding the prediction for the given values
'''
    CI = 50661
        ES = 115641
        AE = 92496
        City = Bengaluru
'''

pred_X = [[1.0 ,0.0, 0.0, 50661,115641,92496]]
pred_Y = regn.predict(pred_X)
print(f"The Predicted Value is : {pred_Y[0]}")
```

```
The Predicted Value is : 90364.23561213724
```

Project 2:

For given “logistic_data.csv” dataset, determine the classification model.

a) Using logistic regression algorithm.

b) Using KNN algorithm

c) Compare (a) and (b) and state which gives better performance in terms of metric parameter such as accuracy score, precision score, recall.

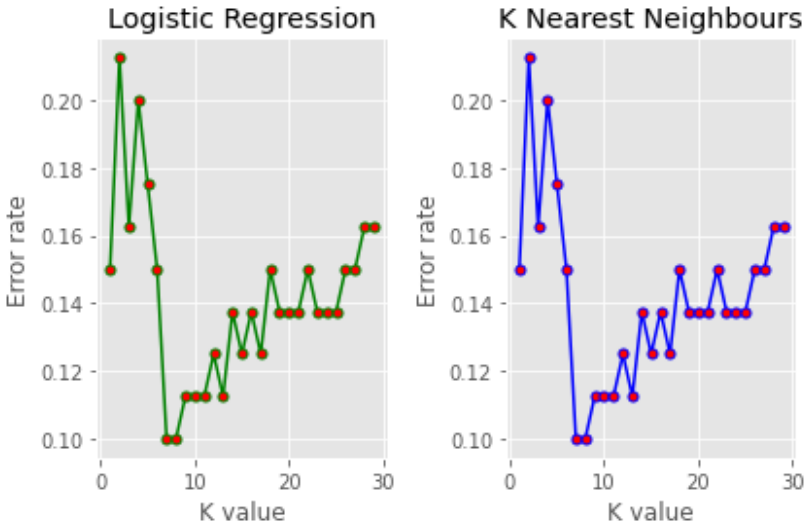
In [2]:

```
'''
Project 2:

For given “logistic_data.csv” dataset, determine the classification model.
a)      Using logistic regression algorithm.
b)      Using KNN algorithm
c)      Compare (a) and (b) and state which gives better performance in terms of metric parameter such as accuracy score, precision score, recall

'''

import numpy as np
import matplotlib.pyplot as plt
plt.style.use('ggplot')
import pandas as pd
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,accuracy_score,precision_score
from sklearn.neighbors import KNeighborsClassifier

# Import data set
dataset=pd.read_csv('../Data/Logistic Data.csv')


# To create feature matrix and dependent variable vector
a=dataset.iloc[:,:-1].values
b=dataset.iloc[:,-1].values

# Replace the missing data
imputer=SimpleImputer(missing_values=np.nan,strategy='mean')
imputer.fit(a[:,:])
a[:,:]=imputer.transform(a[:,:])

# Spiliting of data set into training and testing set
atrain,atest,btrain,btest=train_test_split(a,b,test_size=0.2,random_state=1)

# Feature scaling
sc=StandardScaler()
atrain=sc.fit_transform(atrain)
atest=sc.fit_transform(atest)


# Using logistic regression algorithm.
# Training the classification model
LoR=LogisticRegression(random_state=0)
LoR.fit(atrain,btrain)

# Testing the linear model
```

```
bestimated=LoR.predict(atest)

# Performance matrix
print("Logistic regression :")
print(f"Accuracy score : {accuracy_score(btest,bestimated)}")
print(f"Precision score : {precision_score(btest,bestimated)}")

error_rate_LoR=[]
for i in range(1,30):
    KC=KNeighborsClassifier(n_neighbors=i)
    KC.fit(atrain,btrain)
    bpred_i=KC.predict(atest)
    error_rate_LoR.append(np.mean(bpred_i!=btest))


# By using KNN Algorithm
# Build my KNN classification model
# Training the classification model
KC=KNeighborsClassifier(n_neighbors=7,weights='uniform',p=2)
KC.fit(atrain,btrain)

# Testing the linear model
bestimated=KC.predict(atest)

# Performance matrix
print("\nK Nearest Neighbours : ")
print(f"Accuracy score : {accuracy_score(btest,bestimated)}")
print(f"Precision score : {precision_score(btest,bestimated)}")
print("\n\n")

error_rate_KNN=[]
for i in range(1,30):
    KC=KNeighborsClassifier(n_neighbors=i)
    KC.fit(atrain,btrain)
    bpred_i=KC.predict(atest)
    error_rate_KNN.append(np.mean(bpred_i!=btest))


# Plotting the Data
# Plotting the Error Graph of both the Algorithms
fig = plt.figure()
ax1 = fig.add_subplot(121)
ax2 = fig.add_subplot(122)

ax1.set_title("Logistic Regression")
ax1.plot(range(1,30),error_rate_LoR,marker='o',markerfacecolor='red',markersize=5,color = 'green')
ax1.set_xlabel('K value')
ax1.set_ylabel('Error rate')

ax2.set_title("K Nearest Neighbours")
ax2.plot(range(1,30),error_rate_KNN,marker='o',markerfacecolor='red',markersize=5,color = 'b')
```

```
ax2.set_xlabel('K value')
ax2.set_ylabel('Error rate')

fig.tight_layout()
plt.show()



print('''

According to the performance metrics, The KNN algorithm provides a better accuracy score
and precision score than that of Logistic Regression .

Logistic regression :
Accuracy score : 0.8125
Precision score : 0.8148148148148148

K Nearest Neighbours :
Accuracy score : 0.9
Precision score : 0.8333333333333334

''')
```

```
Logistic regression :
Accuracy score : 0.8125
Precision score : 0.8148148148148148

K Nearest Neighbours :
Accuracy score : 0.9
Precision score : 0.8333333333333334
```



```
According to the performance metrics, The KNN algorithm provides a better accuracy score
and precision score than that of Logistic Regression .

Logistic regression :
Accuracy score : 0.8125
Precision score : 0.8148148148148148

K Nearest Neighbours :
Accuracy score : 0.9
Precision score : 0.8333333333333334
```

Project 3:

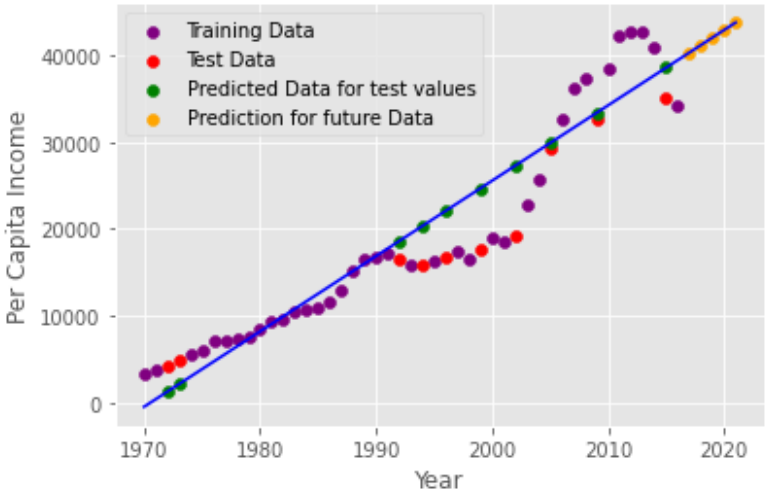
Using captia_income data set, build a regression model and predict the per capitia income for Canadian citizens in year 2021.

In [3]:

```
'''
Using captia_income data set, build a regression model and predict the per capitia income for Canadian citizens in year 2021.

'''

import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder,StandardScaler,Normalizer
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from matplotlib import pyplot as plt
plt.style.use('ggplot')

# Import data set
dataset=pd.read_csv('../Data/captia_income.csv')


# To create feature matrix and dependent variable vector
# X is the year
# Y is the per capita income of that year
x=np.array(dataset.iloc[:,:-1].values)
y=np.array(dataset.iloc[:,-1].values)

# Replace missing data
```

```
imputer=SimpleImputer(missing_values=np.nan,strategy='mean')
imputer.fit(x)
x=imputer.transform(x)


# Splitting of data into training data set and testing data set
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=1)


# Build a multiple linear model
regn=LinearRegression()
regn.fit(xtrain,ytrain)


#Finding the predicted values for the test data
yestimated = regn.predict(xtest)


# Find the predicted per capita income for year 2021
pred_X = [[2017],[2018],[2019],[2020],[2021]]
pred_Y = regn.predict(pred_X)
print(f"The Predicted Value is : {pred_Y[-1]}")

# Plotting the Data
# Plotting the predicted data(green) vs the testing data(red)
plt.xlabel("Year")
plt.ylabel("Per Capita Income")
train_plot = plt.scatter(xtrain,ytrain,color = 'purple')
test_plot = plt.scatter(xtest,ytest,color = 'red')
pred_plot = plt.scatter(xtest,yestimated,color = 'green')
out_plot = plt.scatter(pred_X,pred_Y,color = 'orange')
plt.plot(np.linspace(x[0],2021,len(x)) , regn.intercept_ + regn.coef_* np.linspace(x[0],2021,len(x)),color = "blue")
plt.legend((train_plot,test_plot,pred_plot,out_plot),("Training Data","Test Data","Predicted Data for test values","Prediction for future Data"),
           scatterpoints=1,
           loc='upper left',
           ncol=1,
           fontsize=10)
plt.show()
```

```
The Predicted Value is : 43792.07972987788
```


Training Data
Test Data
Predicted Data for test values
Prediction for future Data
Per Capita Income
Year
0
10000
20000
30000
40000
1970
1980
1990
2000
2010
2020