week6

September 25, 2024

[1]:

```

```

[1]:

```python
import pandas as pd
credit_df = pd.read_csv("C:/Users/HP/Downloads/credit.csv")
credit_df
```

[1]:

```
      id   Income  Limit  Rating  Cards  Age  Education  Gender Student  \
0      1   14.891   3606     283      2   34         11    Male      No
1      2  106.025   6645     483      3   82         15  Female     Yes
2      3  104.593   7075     514      4   71         11    Male      No
3      4  148.924   9504     681      3   36         11  Female      No
4      5   55.882   4897     357      2   68         16    Male      No
..   …       …     …      …     …   …        …       …
395  396   12.096   4100     307      3   32         13    Male      No
396  397   13.364   3838     296      5   65         17    Male      No
397  398   57.872   4171     321      5   67         12  Female      No
398  399   37.728   2525     192      1   44         13    Male      No
399  400   18.701   5524     415      5   64          7  Female      No

    Married          Ethnicity  Balance
0       Yes          Caucasian      333
1       Yes              Asian      903
2        No              Asian      580
3        No              Asian      964
4       Yes          Caucasian      331
..      …                …      …
395     Yes          Caucasian      560
396      No   African American      480
397     Yes          Caucasian      138
398     Yes          Caucasian        0
399      No              Asian      966

[400 rows x 12 columns]
```

[6]:

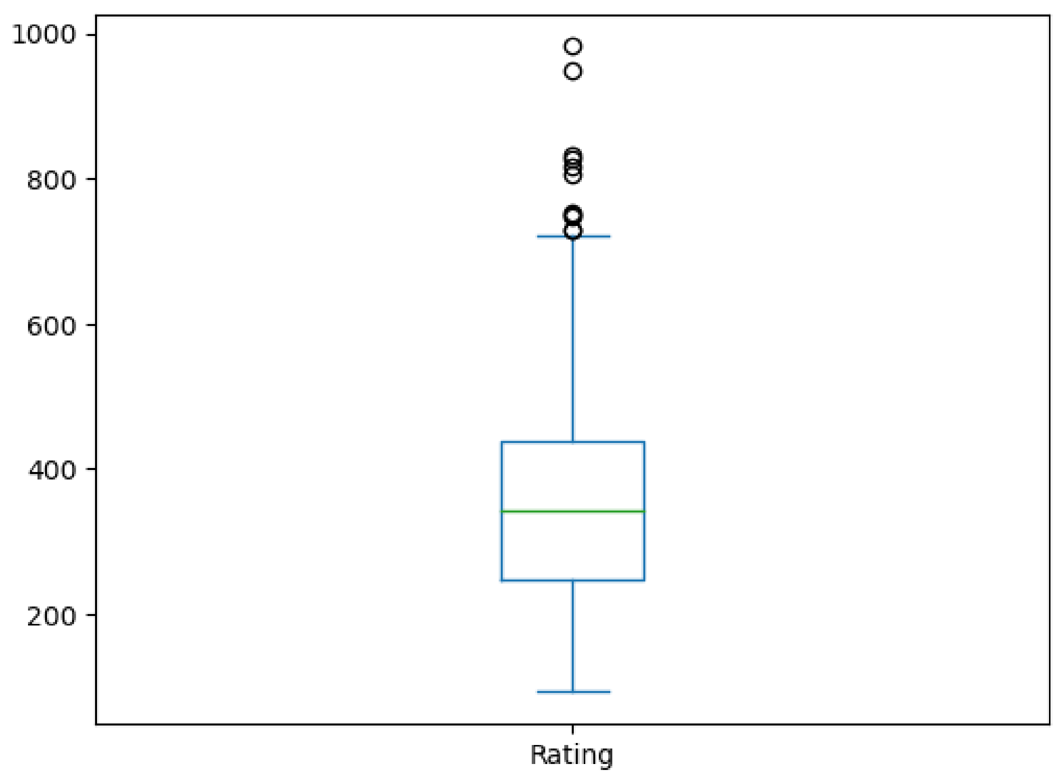
```python
credit_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
```

```
Data columns (total 12 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   id         400 non-null    int64  
 1   Income     400 non-null    float64
 2   Limit      400 non-null    int64  
 3   Rating     400 non-null    int64  
 4   Cards      400 non-null    int64  
 5   Age        400 non-null    int64  
 6   Education  400 non-null    int64  
 7   Gender     400 non-null    object 
 8   Student    400 non-null    object 
 9   Married    400 non-null    object 
 10  Ethnicity  400 non-null    object 
 11  Balance    400 non-null    int64  
dtypes: float64(1), int64(7), object(4)
memory usage: 37.6+ KB
```

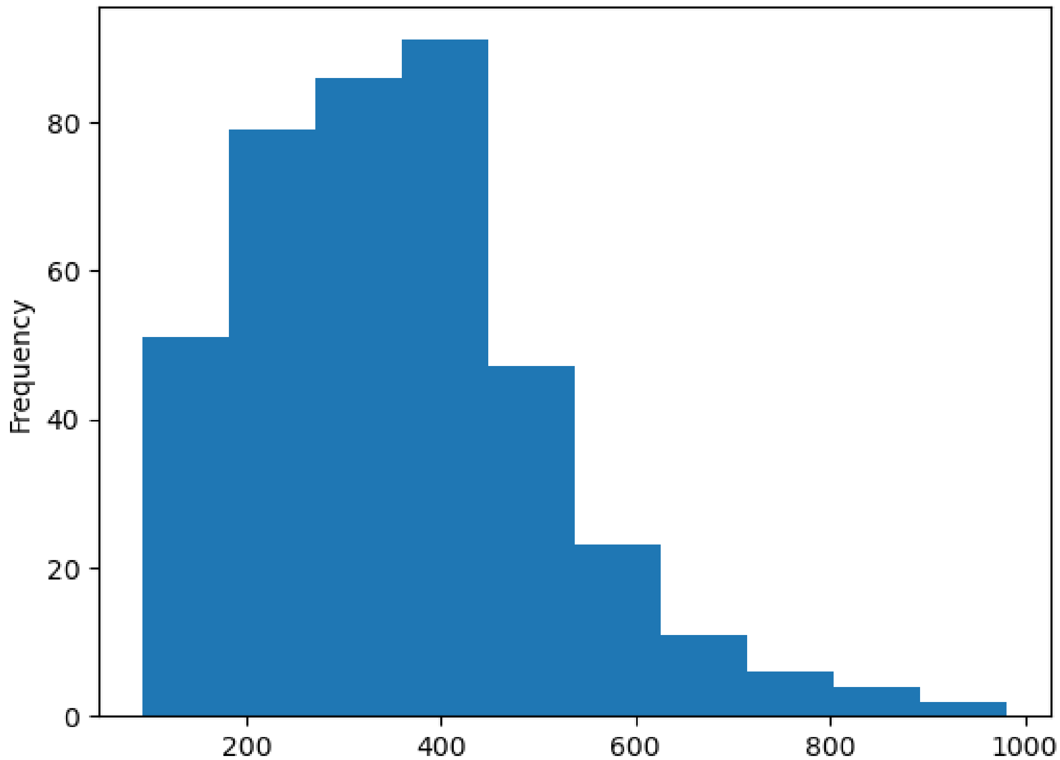
[5]:
```
credit_df['Rating'].plot(kind='box')
```

[5]: <Axes: >



[7]:
```
credit_df['Rating'].plot(kind='hist')
```

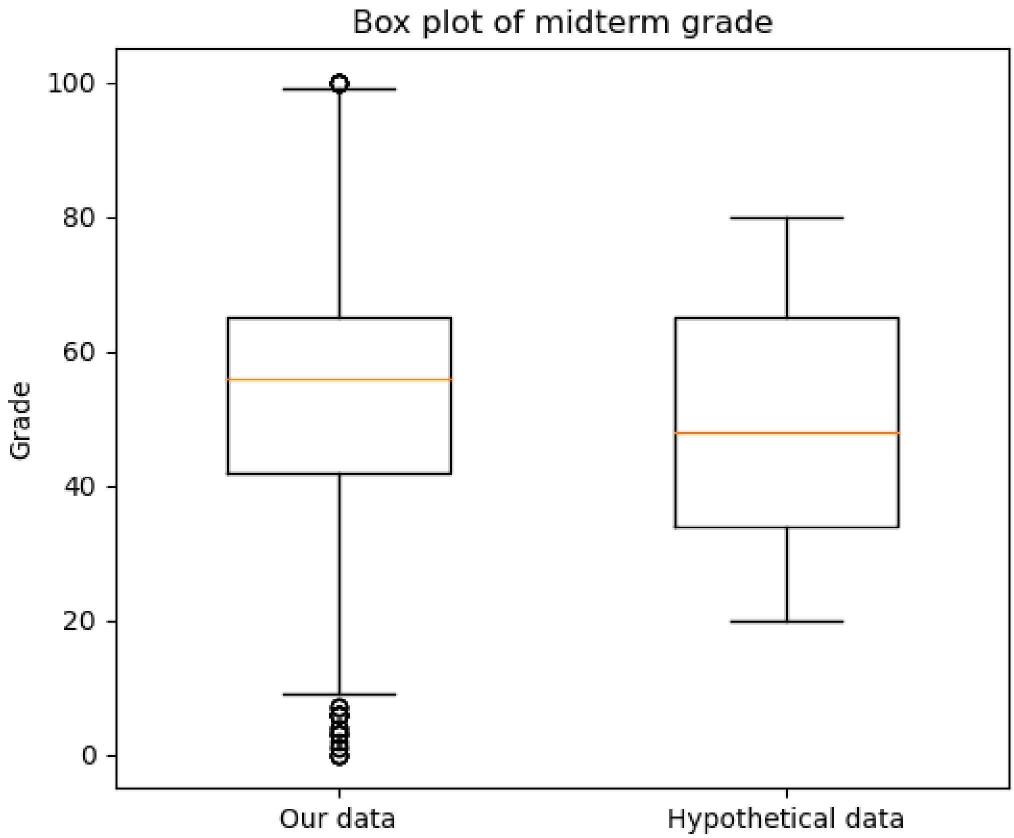
[7]: <Axes: ylabel='Frequency'>



[8]:
```
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(102)
grades = np.concatenate([[50,52,53,55,56,60,61,62,65,67]*20,
np.random.randint(0, 101, size=300)])
Q1 = np.percentile(grades , 25)
Q3 = np.percentile(grades , 75)
Q1,Q3 = np.percentile(grades , [25,75])
IQR = Q3 - Q1
ul = Q3+1.5*IQR
ll = Q1-1.5*IQR
outliers = grades[(grades > ul) | (grades < ll)]
print(outliers)
fig = plt.figure(figsize=(6,5))
hypo = np.random.randint(20, 81, size=500)
plt.boxplot([grades, hypo], widths=0.5)
plt.xticks([1,2],['Our data', 'Hypothetical data'])
plt.ylabel('Grade')
```

```
plt.title('Box plot of midterm grade')
plt.show()
```

```
[  0    7    4    3    0    4    2    7    6  100    1    3    0    3  100  100  100  100
   4    0    3    6    6    6  100    7    6  100  100    6    3    6    1    6    0]
```



[11]:
```
import numpy as np

data = [1, 2, 2, 2, 3, 1, 1, 15, 2, 2, 2, 3, 1, 1, 2]
mean = np.mean(data)
std = np.std(data)

print('Mean of the dataset is:', mean)
print('Standard deviation is:', std)

threshold = 3
outliers = []

for i in data:
    z = (i - mean)- / std
```

```
    if abs(z) > threshold:
        outliers.append(i)

print('Outliers in dataset based on Z-score are:', outliers)
```

```
Mean of the dataset is: 2.6666666666666665
Standard deviation is: 3.3598941782277745
Outliers in dataset based on Z-score are: [15]
```

[12]:
```
q1 = credit_df["Age"].quantile(0.25)
q3 = credit_df['Age'].quantile(0.75)
iqr = q3-q1
upper_bound = q3+(1.5*iqr)
lower_bound = q1-(1.5*iqr)
```

[14]:
```
upperIndex = credit_df[credit_df['Age']>upper_bound].index
credit_df.drop(upperIndex,inplace=True)
lowerIndex = credit_df[credit_df['Age']<lower_bound].index
credit_df.drop(lowerIndex,inplace=True)
credit_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Unnamed: 0   400 non-null    int64
 1   Income       400 non-null    float64
 2   Limit        400 non-null    int64
 3   Rating       400 non-null    int64
 4   Cards        400 non-null    int64
 5   Age          400 non-null    int64
 6   Education    400 non-null    int64
 7   Gender       400 non-null    object
 8   Student      400 non-null    object
 9   Married      400 non-null    object
 10  Ethnicity    400 non-null    object
 11  Balance      400 non-null    int64
dtypes: float64(1), int64(7), object(4)
memory usage: 37.6+ KB
```

[16]:
```
m = np.mean(credit_df['Age'])
print('mean:',m)
for i in credit_df['Age']:
 if i<lower_bound or i>upper_bound :
    titanic_df['Age'] = titanic_df['Age'].replace(i,m)
```

```
mean: 55.6675
```

[17]:
```
m = credit_df['Age'].median()
print("median",m)
for i in credit_df['Age']:
 if i<lower_bound or i>upper_bound :
     credit_df['Age'] = credit_df['Age'].replace(i,m)
```

```
median 56.0
```

[18]:
```
for i in credit_df['Age']:
 if i<lower_bound or i>upper_bound :
     credit_df['Age'] = credit_df['Age'].replace(i,0)
```

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

[5]:
```
card_approval_df=pd.read_csv("C:/Users/HP/Downloads/credit.csv")
print(card_approval_df.head())
```

```
   Unnamed: 0   Income  Limit  Rating  Cards  Age  Education  Gender Student  \
0           1   14.891   3606     283      2   34         11    Male      No
1           2  106.025   6645     483      3   82         15  Female     Yes
2           3  104.593   7075     514      4   71         11    Male      No
3           4  148.924   9504     681      3   36         11  Female      No
4           5   55.882   4897     357      2   68         16    Male      No

  Married  Ethnicity  Balance
0     Yes  Caucasian      333
1     Yes      Asian      903
2      No      Asian      580
3      No      Asian      964
4     Yes  Caucasian      331
```

[6]:
```
print(card_approval_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  400 non-null    int64
 1   Income      400 non-null    float64
 2   Limit       400 non-null    int64
 3   Rating      400 non-null    int64
 4   Cards       400 non-null    int64
 5   Age         400 non-null    int64
```

```
 6   Education   400 non-null    int64
 7   Gender      400 non-null    object
 8   Student     400 non-null    object
 9   Married     400 non-null    object
 10  Ethnicity   400 non-null    object
 11  Balance     400 non-null    int64
dtypes: float64(1), int64(7), object(4)
memory usage: 37.6+ KB
None
```

[7]:

```

```

[7]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
get_ipython().run_line_magic('matplotlib', 'inline')
```

[8]:

```
boston_dataset = pd.read_csv('C:/Users/HP/Downloads/BostonHousing.csv')
boston_dataset.keys()
```

[8]:

```
Index(['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax',
       'ptratio', 'b', 'lstat', 'medv'],
      dtype='object')
```

[9]:

```
boston_dataset.head(5)
```

[9]:

```
      crim    zn  indus  chas    nox     rm   age     dis  rad  tax  ptratio  \
0  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296     15.3
1  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242     17.8
2  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242     17.8
3  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222     18.7
4  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222     18.7

        b  lstat  medv
0  396.90   4.98  24.0
1  396.90   9.14  21.6
2  392.83   4.03  34.7
3  394.63   2.94  33.4
4  396.90   5.33  36.2
```

[10]:

```
boston_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   crim     506 non-null    float64
```

```
 1   zn       506 non-null    float64
 2   indus    506 non-null    float64
 3   chas     506 non-null    int64
 4   nox      506 non-null    float64
 5   rm       506 non-null    float64
 6   age      506 non-null    float64
 7   dis      506 non-null    float64
 8   rad      506 non-null    int64
 9   tax      506 non-null    int64
 10  ptratio  506 non-null    float64
 11  b        506 non-null    float64
 12  lstat    506 non-null    float64
 13  medv     506 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 55.5 KB
```

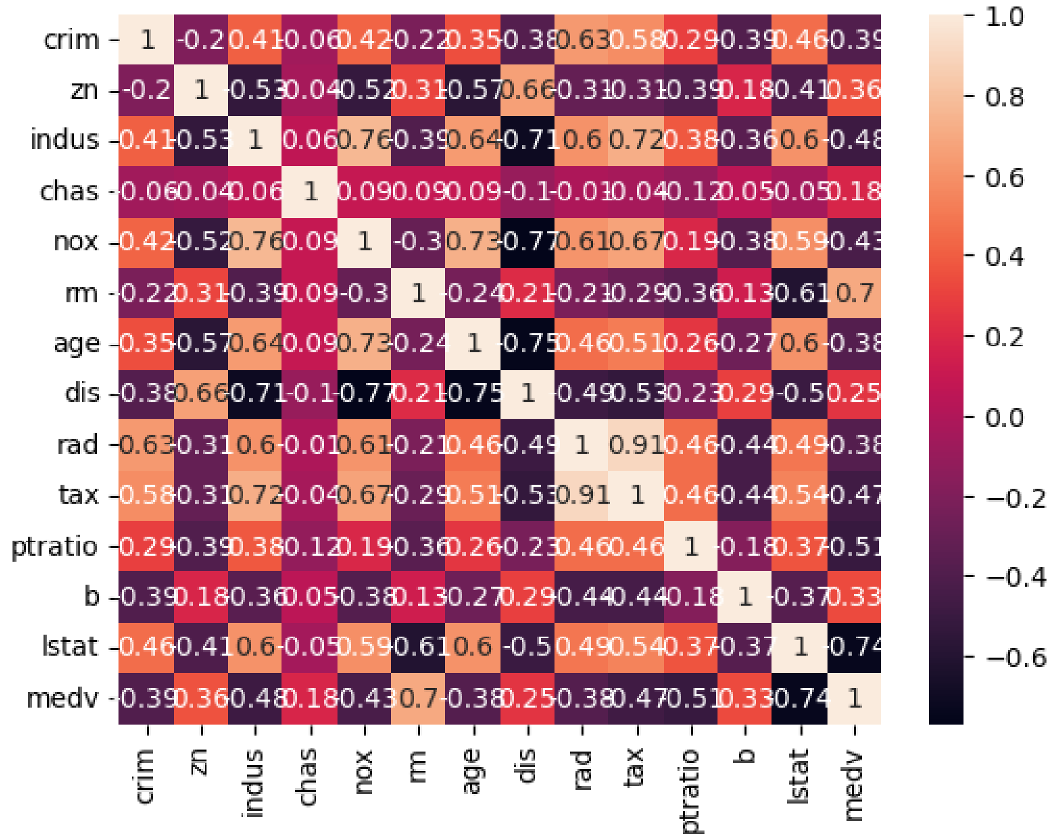
[11]:
```
boston_dataset.isnull().sum()
```

[11]:
```
crim       0
zn         0
indus      0
chas       0
nox        0
rm         0
age        0
dis        0
rad        0
tax        0
ptratio    0
b          0
lstat      0
medv       0
dtype: int64
```

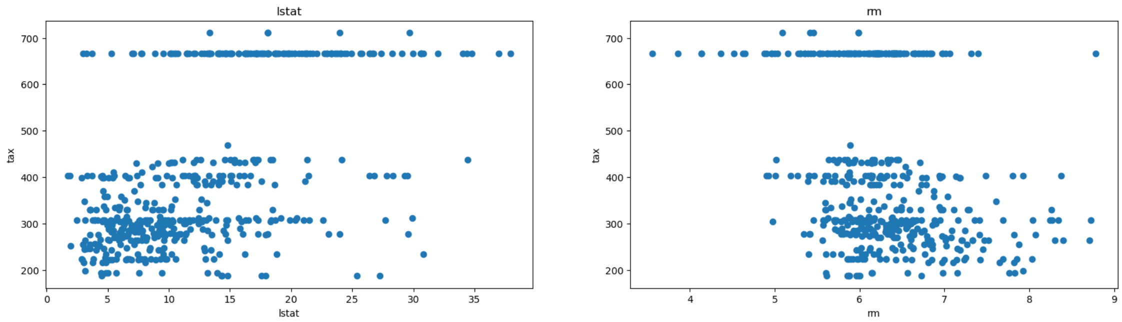
[12]:
```
correlation_matrix = boston_dataset.corr().round(2)
```

[13]:
```
sns.heatmap(data=correlation_matrix, annot=True)
```

[13]:
```
<Axes: >
```



[55]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(20, 5))
features = ['lstat', 'rm']
target = boston_dataset['tax']
for i, col in enumerate(features):
    plt.subplot(1, len(features) , i+1)
    x = boston_dataset[col]
    y = target
    plt.scatter(x, y, marker='o')
    plt.title(col)
    plt.xlabel(col)
    plt.ylabel('tax')
```



[16]:
```
X = pd.DataFrame(np.c_[boston_dataset['tax'], boston_dataset['rm']],␣
 ↪columns=['lstat', 'rm'])
Y = boston_dataset['age']
```

[17]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,␣
 ↪random_state=42)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(404, 2)
(102, 2)
(404,)
(102,)
```

[18]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

[19]:
```
lin_model = LinearRegression()
lin_model.fit(X_train, Y_train)
```

[19]: LinearRegression()

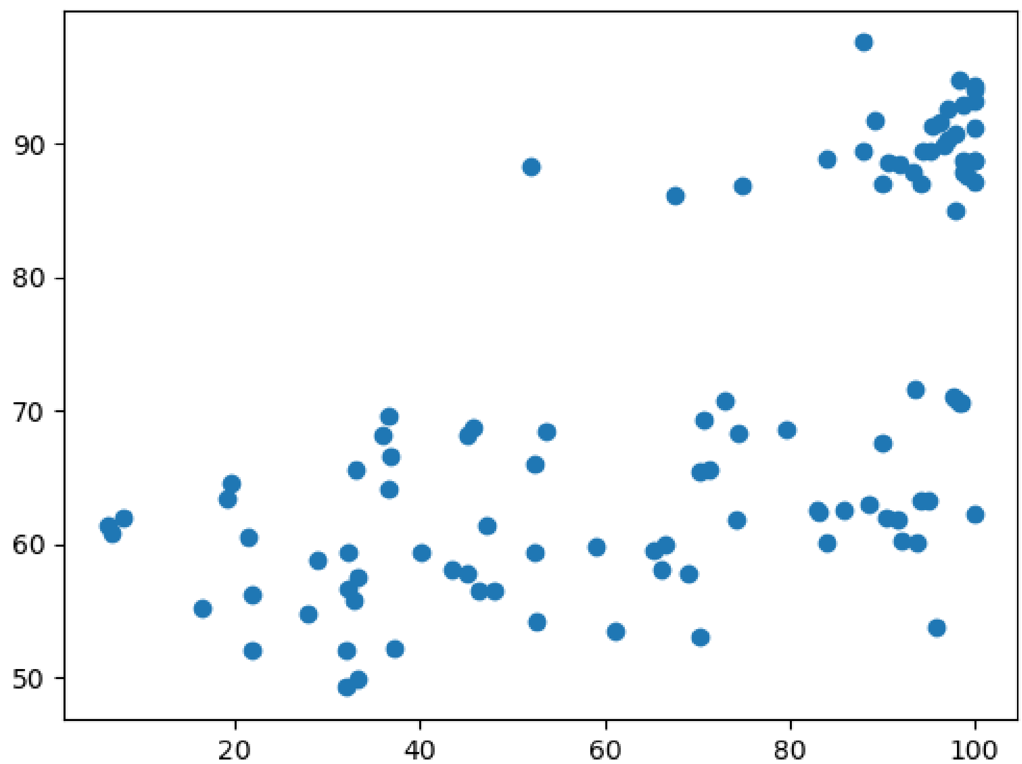
[21]:
```
y_train_predict = lin_model.predict(X_train)
rmse = (np.sqrt(mean_squared_error(Y_train, y_train_predict)))
r2 = r2_score(Y_train, y_train_predict)
print("The model performance for training set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
print("\n")
# model evaluation for testing set
y_test_predict = lin_model.predict(X_test)
```

```
# root mean square error of the model
rmse = (np.sqrt(mean_squared_error(Y_test, y_test_predict)))
# r-squared score of the model
r2 = r2_score(Y_test, y_test_predict)
print("The model performance for testing set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
```

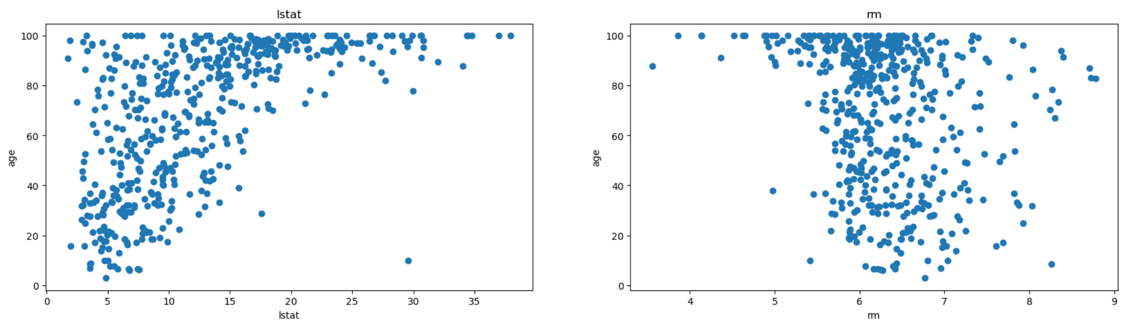
```
The model performance for training set
--------------------------------------
RMSE is 24.54898757784132
R2 score is 0.2291232217092818


The model performance for testing set
--------------------------------------
RMSE is 22.33059428323184
R2 score is 0.39666537877593555
```

[22]:
```
plt.scatter(Y_test, y_test_predict)
plt.show()
```



[24]:
```
plt.figure(figsize=(20, 5))
features = ['lstat', 'rm']
target = boston_dataset['age']
for i, col in enumerate(features):
    plt.subplot(1, len(features) , i+1)
    x = boston_dataset[col]
    y = target
    plt.scatter(x, y, marker='o')
    plt.title(col)
    plt.xlabel(col)
    plt.ylabel('age')
```

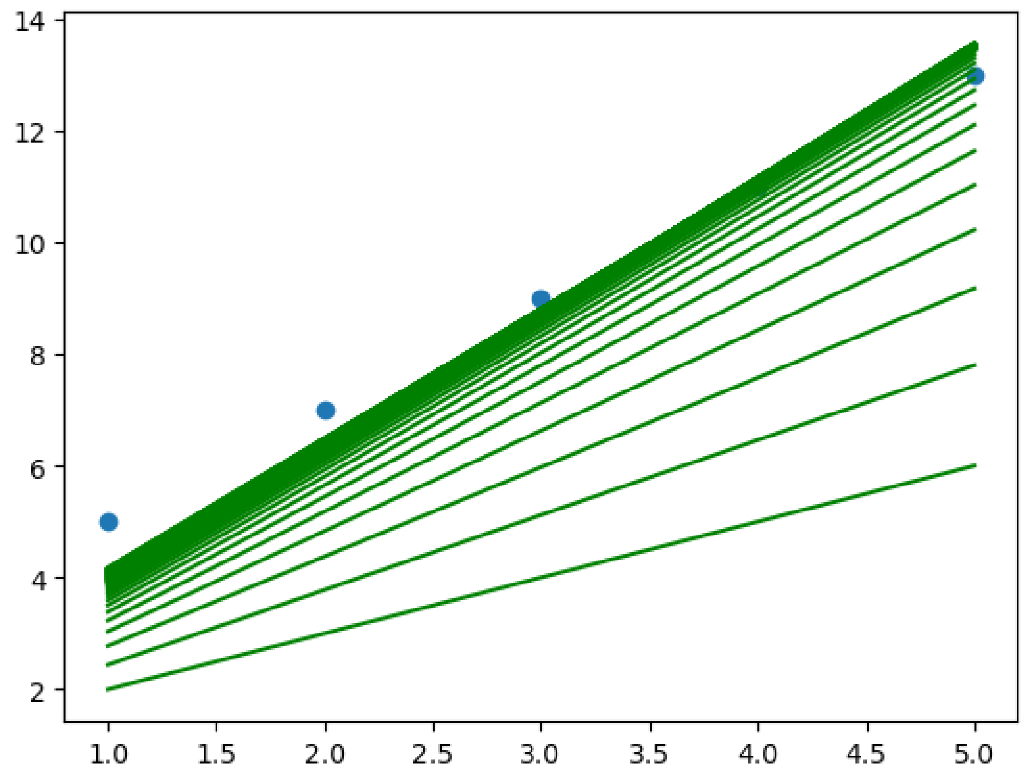


[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[28]:
```
%matplotlib inline
def gradient_descent(x,y):
    m = b = 1
    rate = 0.01
    n = len(x)
    plt.scatter(x,y)
    for i in range(100):
        y_predicted = m * x + b
        plt.plot(x,y_predicted,color='green')
        md = -(2/n)*sum(x*(y-y_predicted))
        yd = -(2/n)*sum(y-y_predicted)
        m = m - rate * md
        b = b - rate * yd
```

[29]:
```
x = np.array([1,2,3,4,5])
y = np.array([5,7,9,11,13])
```

[30]:
```
gradient_descent(x,y)
```



[31]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Importing the dataset
datas = pd.read_csv('C:/Users/HP/Downloads/BostonHousing.csv')
datas
```

[31]:
```
        crim    zn  indus  chas    nox     rm   age     dis  rad  tax  \
0    0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296
1    0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242
2    0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242
3    0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222
4    0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222
..       …     …      …     …      …      …     …      …   …   …
501  0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273
502  0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273
503  0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273
504  0.10959   0.0  11.93     0  0.573  6.794  89.3  2.3889    1  273
505  0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273

     ptratio       b  lstat  medv
0       15.3  396.90   4.98  24.0
```

```
1       17.8  396.90   9.14  21.6
2       17.8  392.83   4.03  34.7
3       18.7  394.63   2.94  33.4
4       18.7  396.90   5.33  36.2
..       …      …     …     …
501     21.0  391.99   9.67  22.4
502     21.0  396.90   9.08  20.6
503     21.0  396.90   5.64  23.9
504     21.0  393.45   6.48  22.0
505     21.0  396.90   7.88  11.9

[506 rows x 14 columns]
```

[33]:
```
X = datas.iloc[:, 1:2].values
y = datas.iloc[:, 2].values
```

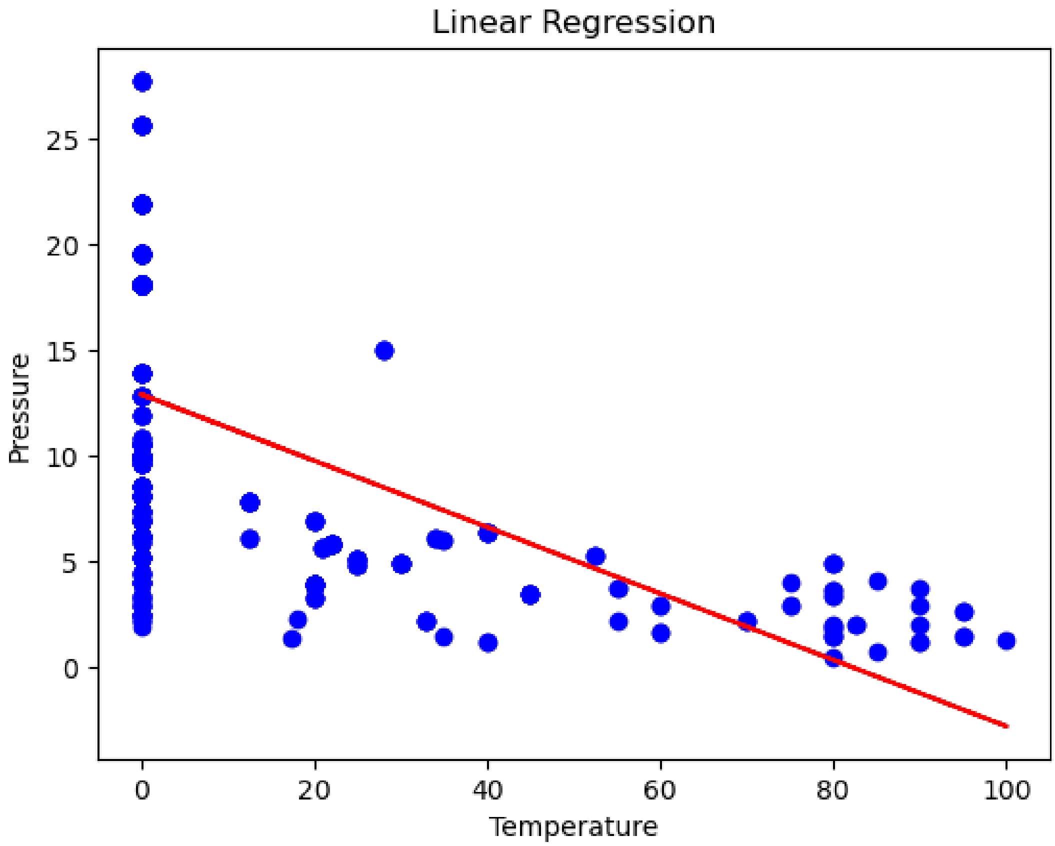
[34]:
```
from sklearn.linear_model import LinearRegression
lin = LinearRegression()
lin.fit(X, y)
```

[34]: LinearRegression()

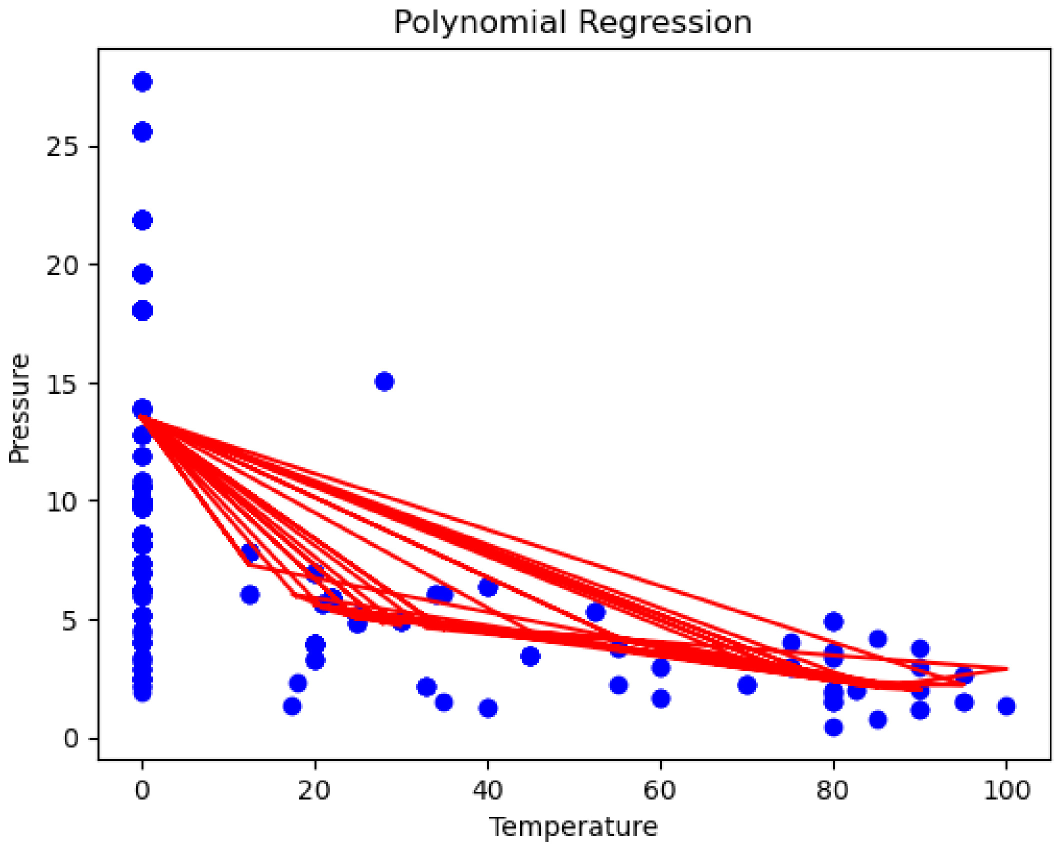
[35]:
```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree = 4)+
X_poly = poly.fit_transform(X)
poly.fit(X_poly, y)
lin2 = LinearRegression()
lin2.fit(X_poly, y)
```

[35]: LinearRegression()

[36]:
```
plt.scatter(X, y, color = 'blue')
plt.plot(X, lin.predict(X), color = 'red')
plt.title('Linear Regression')
plt.xlabel('Temperature')
plt.ylabel('Pressure')
plt.show(
```



[37]:
```
plt.scatter(X, y, color = 'blue')
plt.plot(X, lin2.predict(poly.fit_transform(X)), color = 'red')
plt.title('Polynomial Regression')
plt.xlabel('Temperature')
plt.ylabel('Pressure')
plt.show()
```



[38]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,r2_score
```

[39]:
```
from sklearn.model_selection import train_test_split
```

[42]:
```
df = pd.read_csv('C:/Users/HP/Downloads/Advertising.csv.')
df
```

[42]:

	ID	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9
..	…	…	…	…	…
195	196	38.2	3.7	13.8	7.6
196	197	94.2	4.9	8.1	9.7

```
197  198  177.0    9.3        6.4   12.8
198  199  283.6   42.0       66.2   25.5
199  200  232.1    8.6        8.7   13.4

[200 rows x 5 columns]
```

[43]:
```
df.dropna(inplace=True,axis=0)
df
```

[43]:
```
      ID     TV  Radio  Newspaper  Sales
0      1  230.1   37.8       69.2   22.1
1      2   44.5   39.3       45.1   10.4
2      3   17.2   45.9       69.3    9.3
3      4  151.5   41.3       58.5   18.5
4      5  180.8   10.8       58.4   12.9
..   …    …    …         …     …
195  196   38.2    3.7       13.8    7.6
196  197   94.2    4.9        8.1    9.7
197  198  177.0    9.3        6.4   12.8
198  199  283.6   42.0       66.2   25.5
199  200  232.1    8.6        8.7   13.4

[200 rows x 5 columns]
```

[45]:
```
y = df['TV']
X = df.drop('TV',axis=1)
```

[46]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.
 ↪3,random_state=101)
```

[47]:
```
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

[48]:
```
lr = LinearRegression()
model = lr.fit(X_train,y_train)
```

[49]:
```
y_pred = model.predict(X_test)
ydf = pd.DataFrame({'y_test':y_test,'y_pred':y_pred})
rslt_df = ydf.sort_values(by = 'y_test')
```

[50]:
```
print(mean_squared_error(y_test,y_pred))
```

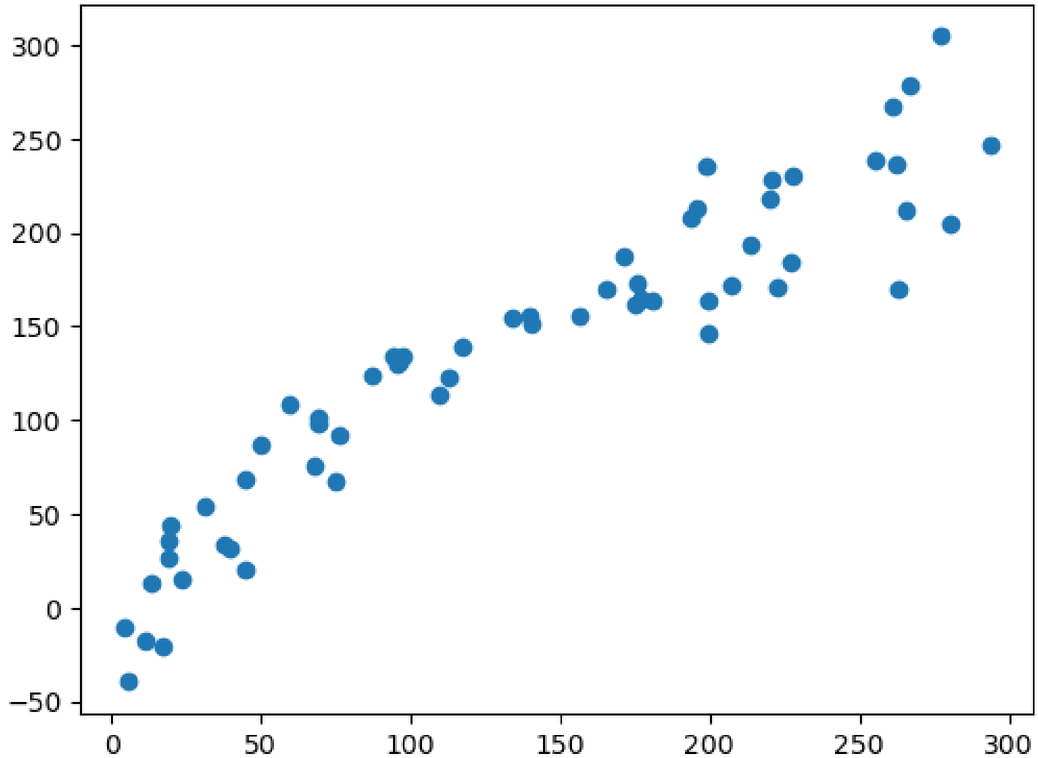
```
933.1209747971208
```

[51]:
```
print(r2_score(y_test, y_pred))
```

```
0.8790644106224436
```

[52]:
```python
import matplotlib.pyplot as plt
plt.scatter(ydf['y_test'],ydf['y_pred'])
```

[52]: <matplotlib.collections.PathCollection at 0x20664920ad0>



[53]:
```python
model.coef_
```

[53]: array([4.19811629, -49.89190342, 2.02818532, 94.1700075])

[54]:
```python
model.intercept_
```

[54]: 151.66071428571433

[7]:
```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Load the dataset from a file
# Replace 'your_dataset.csv' with the actual path to your file
data = pd.read_csv("C:/Users/HP/Downloads/credit.csv")
```

```python
bus = pd.read_csv("C:/Users/HP/Downloads/marksheet.csv")
# Assuming the dataset has columns 'feature1', 'feature2', ..., 'target'
# Replace these with the actual column names
X = data[['Education', 'Balance', 'Rating']]  # Feature columns
Y = data['Income']  # Target column

# Split the dataset into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,␣
 ↪random_state=42)

# Train the Linear Regression model
lin_model = LinearRegression()
lin_model.fit(X_train, Y_train)

# Model evaluation for training set
y_train_predict = lin_model.predict(X_train)
rmse_train = np.sqrt(mean_squared_error(Y_train, y_train_predict))
r2_train = r2_score(Y_train, y_train_predict)

print("The model performance for training set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse_train))
print('R2 score is {}'.format(r2_train))
print("\n")

# Model evaluation for testing set
y_test_predict = lin_model.predict(X_test)
rmse_test = np.sqrt(mean_squared_error(Y_test, y_test_predict))
r2_test = r2_score(Y_test, y_test_predict)

print("The model performance for testing set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse_test))
print('R2 score is {}'.format(r2_test))
```

```
The model performance for training set
--------------------------------------
RMSE is 15.02117077384028
R2 score is 0.8185903172179333


The model performance for testing set
--------------------------------------
RMSE is 15.36793364811597
R2 score is 0.8064320939429278
```

[12]:
```python
credit_df = pd.read_csv("C:/Users/HP/Downloads/credit.csv")
credit_df
```

[12]:
```
     Unnamed: 0   Income  Limit  Rating  Cards  Age  Education  Gender  \
0             1   14.891   3606     283      2   34         11    Male
1             2  106.025   6645     483      3   82         15  Female
2             3  104.593   7075     514      4   71         11    Male
3             4  148.924   9504     681      3   36         11  Female
4             5   55.882   4897     357      2   68         16    Male
..          …       …     …     …    …  …         …      …
395         396   12.096   4100     307      3   32         13    Male
396         397   13.364   3838     296      5   65         17    Male
397         398   57.872   4171     321      5   67         12  Female
398         399   37.728   2525     192      1   44         13    Male
399         400   18.701   5524     415      5   64          7  Female

    Student Married         Ethnicity  Balance
0        No     Yes         Caucasian      333
1       Yes     Yes             Asian      903
2        No      No             Asian      580
3        No      No             Asian      964
4        No     Yes         Caucasian      331
..      …     …               …      …
395      No     Yes         Caucasian      560
396      No      No  African American      480
397      No     Yes         Caucasian      138
398      No     Yes         Caucasian        0
399      No      No             Asian      966

[400 rows x 12 columns]
```

[13]:
```python
credit_df = pd.read_csv("C:/Users/HP/Downloads/marksheet.csv")
credit_df
```

[13]:
```
      id      Name  Gender  Age Section  Science  English  History  Maths
0      1   Bronnie  Female   13       C       21       81       62     49
1      2    Lemmie    Male   15       B       29       41       17     40
2      3     Danya  Female   14       C       12       87       16     96
3      4     Denna  Female   14       B       15       53       82     33
4      5   Jocelin    Male   14       A       43        6        3     21
..   …       …     …  …     …      …      …      …    …
245  246    Nickie    Male   13       C       28       15       25     10
246  247       Rog  Female   13       B        1        4       68     65
247  248      Kaia    Male   15       B       93       48       82     44
248  249      Anni  Female   14       B       35       73       66     59
249  250  Fernande    Male   15       B       50        8       80     78
```

```
[250 rows x 9 columns]
```

[19]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Load the datasets
data_credit = pd.read_csv("C:/Users/HP/Downloads/credit.csv")
data_marksheet = pd.read_csv("C:/Users/HP/Downloads/marksheet.csv")

# Display the first few rows of each dataset to understand their structure
print("Credit Dataset:")
print(data_credit.head())
print("\nMarksheet Dataset:")
print(data_marksheet.head())

# For the sake of this example, let's assume both datasets share a common␣
↪column 'ID'
# and we want to predict 'Income' from the credit dataset using features from␣
↪both datasets.

# Merge the datasets on a common column 'ID'
merged_data = pd.merge(data_credit, data_marksheet, on='id')

# Select features from both datasets
X = merged_data[['Education', 'Balance', 'Rating', 'Maths', 'English']]  #␣
↪Example features
Y = merged_data['Income']  # Target variable

# Split the combined dataset into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,␣
↪random_state=42)

# Train the Linear Regression model
lin_model = LinearRegression()
lin_model.fit(X_train, Y_train)

# Model evaluation for training set
y_train_predict = lin_model.predict(X_train)
rmse_train = np.sqrt(mean_squared_error(Y_train, y_train_predict))
r2_train = r2_score(Y_train, y_train_predict)

print("\nThe model performance for the training set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse_train))
```

```
print('R2 score is {}'.format(r2_train))

# Model evaluation for testing set
y_test_predict = lin_model.predict(X_test)
rmse_test = np.sqrt(mean_squared_error(Y_test, y_test_predict))
r2_test = r2_score(Y_test, y_test_predict)

print("\nThe model performance for the testing set")
print("--------------------------------------------")
print('RMSE is {}'.format(rmse_test))
print('R2 score is {}'.format(r2_test))
```

```
Credit Dataset:
   id   Income  Limit  Rating  Cards  Age  Education  Gender Student Married  \
0   1   14.891   3606     283      2   34         11    Male      No     Yes
1   2  106.025   6645     483      3   82         15  Female     Yes     Yes
2   3  104.593   7075     514      4   71         11    Male      No      No
3   4  148.924   9504     681      3   36         11  Female      No      No
4   5   55.882   4897     357      2   68         16    Male      No     Yes

   Ethnicity  Balance
0  Caucasian      333
1      Asian      903
2      Asian      580
3      Asian      964
4  Caucasian      331

Marksheet Dataset:
   id     Name  Gender  Age Section  Science  English  History  Maths
0   1  Bronnie  Female   13       C       21       81       62     49
1   2   Lemmie    Male   15       B       29       41       17     40
2   3    Danya  Female   14       C       12       87       16     96
3   4    Denna  Female   14       B       15       53       82     33
4   5  Jocelin    Male   14       A       43        6        3     21

The model performance for the training set
------------------------------------------
RMSE is 15.284125890822864
R2 score is 0.7910326438413708

The model performance for the testing set
-----------------------------------------
RMSE is 13.685085424490035
R2 score is 0.8125264172047937
```

[15]:
```
credit_df = pd.read_csv("C:/Users/HP/Downloads/credit.csv")
credit_df
```

[15]:
```
      id   Income  Limit  Rating  Cards  Age  Education  Gender Student  \
0      1   14.891   3606     283      2   34         11    Male      No
1      2  106.025   6645     483      3   82         15  Female     Yes
2      3  104.593   7075     514      4   71         11    Male      No
3      4  148.924   9504     681      3   36         11  Female      No
4      5   55.882   4897     357      2   68         16    Male      No
..   …        …      …       …    …    …          …      …       …
395  396   12.096   4100     307      3   32         13    Male      No
396  397   13.364   3838     296      5   65         17    Male      No
397  398   57.872   4171     321      5   67         12  Female      No
398  399   37.728   2525     192      1   44         13    Male      No
399  400   18.701   5524     415      5   64          7  Female      No

    Married         Ethnicity  Balance
0       Yes         Caucasian      333
1       Yes             Asian      903
2        No             Asian      580
3        No             Asian      964
4       Yes         Caucasian      331
..      …                 …      …
395     Yes         Caucasian      560
396      No  African American      480
397     Yes         Caucasian      138
398     Yes         Caucasian        0
399      No             Asian      966

[400 rows x 12 columns]
```

[ ]: