CP8210 Assignment 2

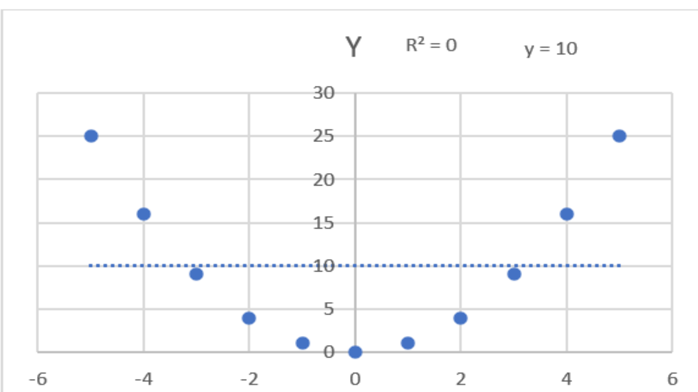
Statistical Models & Regression Analysis

Question#2:

Table 2.1: Dataset

X	Y
-4	16
-2	4
1	1
3	9
-1	1
-5	25
4	16
2	4
0	0
-3	9
5	25

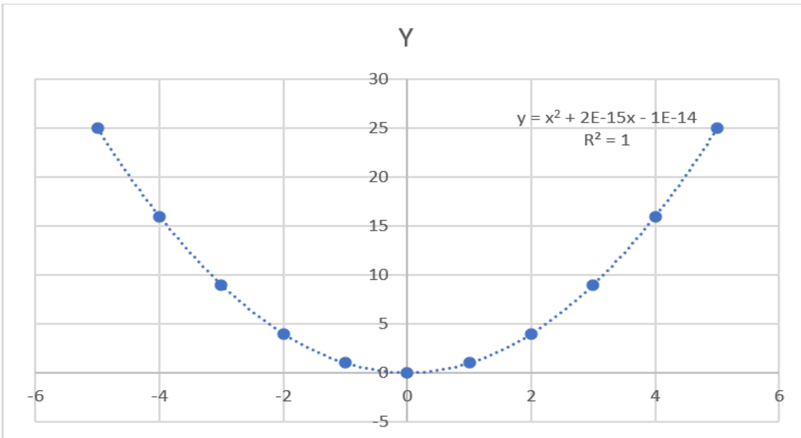
Figure 2.1: Scattered Plot and Linear Regression



2.a. The scatter plot of the data presented in table 1 illustrates that there is no linear relationship between the independent and the dependent variables. Instead, it can be interpreted that there is a strong curvilinear relationship. The correlation of determination R^2 is also equal to 0, which suggests that linear relationship between x and y variables does not exist.

2.b. The estimated linear regression equation for the data presented above is $y = 0x + 10$. As we can see from the figure above, the linear regression model is not the best model for the given data. The quadratic regression model is a better fit for the dataset that is provided and it is also shown in the figure below. The quadratic regression equation is y = x^2 + 2E-15x - 1E-14 and the R^2 now is equal to 1.

Figure 2.2: Scattered Plot and Quadratic Model



2.c. The hypothesis test rests for the slope are shown in the figure below. These results support the null hypothesis that the slope is equal to 0 and therefore the x and y variables have no relationship. The multiple R shown below the regression statistics table in table 2.2 is equal to 0, as well as the R squared and these results support the hypothesis that the slope is equal to 0.

Table 2.2: Hypothesis Test

X	Y
-4	16
-2	4
1	1
3	9
-1	1
-5	25
4	16
2	4
0	0
-3	9
5	25

HA:	B1=0
H0:	B1!=0

X		*Y*	
Mean	0	Mean	10
Standard Error	1	Standard E	2.792848
Median	0	Median	9
Mode	#N/A	Mode	16
Standard Deviatic	3.31662	Standard [	9.2628289
Sample Variance	11	Sample Va	85.8
Kurtosis	-1.2	Kurtosis	-0.9401709
Skewness	0	Skewness	0.6597457
Range	10	Range	25
Minimum	-5	Minimum	0
Maximum	5	Maximum	25
Sum	0	Sum	110
Count	11	Count	11
	0		

	X	*Y*
Column 1	1	
Column 2	0	1

SUMMARY OUTPUT

Regression Statistics	
Multiple R	0
R Square	0
Adjusted R Square	-0.111111111
Standard Error	9.763879011
Observations	11

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	1	0	0	0	1
Residual	9	858	95.3333		
Total	10	858			

	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	10	2.943920289	3.39683	0.00791	3.3403896	16.6596	3.34038963	16.6596104
X Variable 1	0	0.930949336	0	1	-2.1059537	2.10595	-2.1059537	2.10595371

Question#3

Dataset and Exploratory Data Analysis

The heart disease dataset used for this research assignment is obtained from the UCI Machine Learning Repository and in particular the processed cleaveland data is analyzed [1 ,2]. There are a total of 303 samples and the dataset does have missing values as shown in the figure below. From the figure below, it can be seen that out of the 303 samples there are 6 rows with missing values and after dropping these rows the total samples left for analysis are 297.

Figure 3.1: Drop Missing Values

```
[204] #drop samples with missing values
      df.isnull().sum()

      age         0
      sex         0
      cp          0
      testbps     0
      chol        0
      fbs         0
      restecg     0
      thalach     0
      exang       0
      oldpeak     0
      slope       0
      ca          4
      thal        2
      target      0
      dtype: int64

[205] df.shape

      (303, 14)

[206] data=df.dropna()
```

There are 13 different attributes that affect the outcome of the target variable and these attributes are shown in the second figure below.

Figure 3.2: Heart Disease Dataset

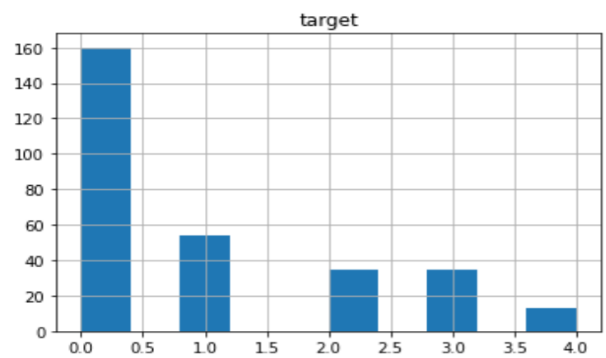
	age	sex	cp	testbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	1	145	233	1	2	150	0	2.3	3	0.0	6.0	0
1	67	1	4	160	286	0	2	108	1	1.5	2	3.0	3.0	2
2	67	1	4	120	229	0	2	129	1	2.6	2	2.0	7.0	1
3	37	1	3	130	250	0	0	187	0	3.5	3	0.0	3.0	0
4	41	0	2	130	204	0	2	172	0	1.4	1	0.0	3.0	0

It can be observed that some of the attributes are categorical including: sex, cp, fbs, restecg, exang, slope, ca, and thal. It is necessary to encode these categorical attributes before the data is used for training. In this assignment the get_dummies function was implemented to encode these attributes. The following attributes: age, testbps, chol, thalach, and oldpeak were standardized using the StandardScaler function. The dataset was also split into two and 80% is used for training and 20% for testing.

Figure 3.3: Data Distribution

```
0    160
1     54
2     35
3     35
4     13
Name: target, dtype: int64
```

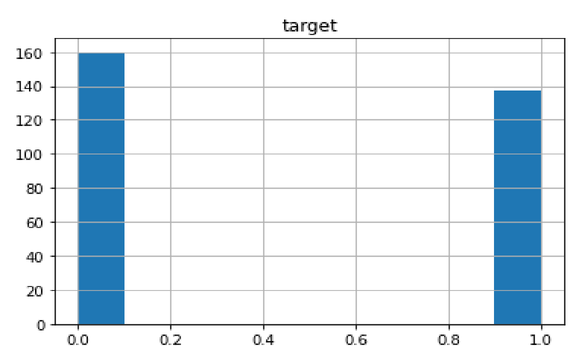
Figure 3.4: Data Distribution Bar Graph



The target variable categorizes the severity of the heart disease where 0 refers to cases where heart disease is not present. It is important to note how unevenly distributed the dataset is and it is expected that the models would perform poorly at recognizing the severity of the heart disease.

If this multi-class classification problem is converted to a binary classification problem and all the values ranging from 1 to 4 are put in the 1 category which indicates presence of heart disease then the problem of uneven distribution can be solved.

Figure 3.5: Binary Data Distribution Bar Graph



In the sections below, the logistic regression model will be implemented to solve the multi-class classification problem as well as the binary classification problem.

Correlation Matrix and Feature Selection:

The correlation matrix of the heart disease dataset is shown in the figures below. There are two provided, the first one shows the correlation of all the 13 attributes, and the second one includes the encoded variables as well. The highly correlated and negatively correlated attributes can be observed and feature selection as well as reduction is also performed to reduce any redundancies during the training of the models. The python code shown below is used for performing the feature selection and identifying positively and negatively correlated features. Typically, an 80% or 90% correlation between two variables indicates strong relationship, but from the first correlation matrix shown in figure 3.8 with the attributes not encoded it can be seen that the connections are not very strong. The strongest positive correlation can be observed between oldpeak and slope which is 0.56 and a negative correlation of -0.41 can be seen for thalach and age. It is a good practice to identify these correlations and if two features are highly or negatively correlated then one feature can be dropped. Doing so avoids any redundancy and any duplicates can also be eliminated. Since the correlation is not very strong in the case where features are not encoded, a percentage of 0.3 is used to get all correlated features. A total of 6 features were obtained after running the code and the following features were dropped: ca, exang, oldpeak, slope, thal, and thalach.

The correlation matrix obtained after encoding the categorical features shows stronger relationship between the attributes and the performance is significantly improved when the encoded features are used for training and testing the models. This is illustrated in figure 3.9. All features with greater than or equal to 0.8 correlation were obtained and these attributes include: exang_1, fbs_1, restecg_2, sex_1, slope_2, and thal_7.0. These features were dropped and 22 features were kept to perform linear and logistic regression.

Figure 3.6: Not Encoded

```
[98] #feature selection
     def correlation(dataset, threshold):
         col_corr = set()
         corr_matrix = dataset.corr()
         for i in range(len(corr_matrix.columns)):
             for j in range(i):
                 if abs(corr_matrix.iloc[i, j]) > threshold:
                     colname = corr_matrix.columns[i]
                     col_corr.add(colname)
         return col_corr
```

```
[100] corr_features = correlation(x_train, 0.3)
      len(set(corr_features))

      6
```

```
[101] corr_features

      {'ca', 'exang', 'oldpeak', 'slope', 'thal', 'thalach'}
```

```
[102] x_train.drop(corr_features,axis=1)
      x_test.drop(corr_features,axis=1)
```

Figure 3.7: Encoded

```
def correlation(dataset, threshold):
    col_corr = set()  # Set of all the names of correlated columns
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold: # we are interested in absolute coeff value
                colname = corr_matrix.columns[i]  # getting the name of column
                col_corr.add(colname)
    return col_corr
```

```
[69] corr_features = correlation(x_train, 0.8)
     len(set(corr_features))

     6
```

```
[70] corr_features

     {'exang_1', 'fbs_1', 'restecg_2', 'sex_1', 'slope_2', 'thal_7.0'}
```

```
[71] x_train.drop(corr_features,axis=1)
     x_test.drop(corr_features,axis=1)
```

Figure 3.8: Correlation Matrix for Dataset Not Encoded

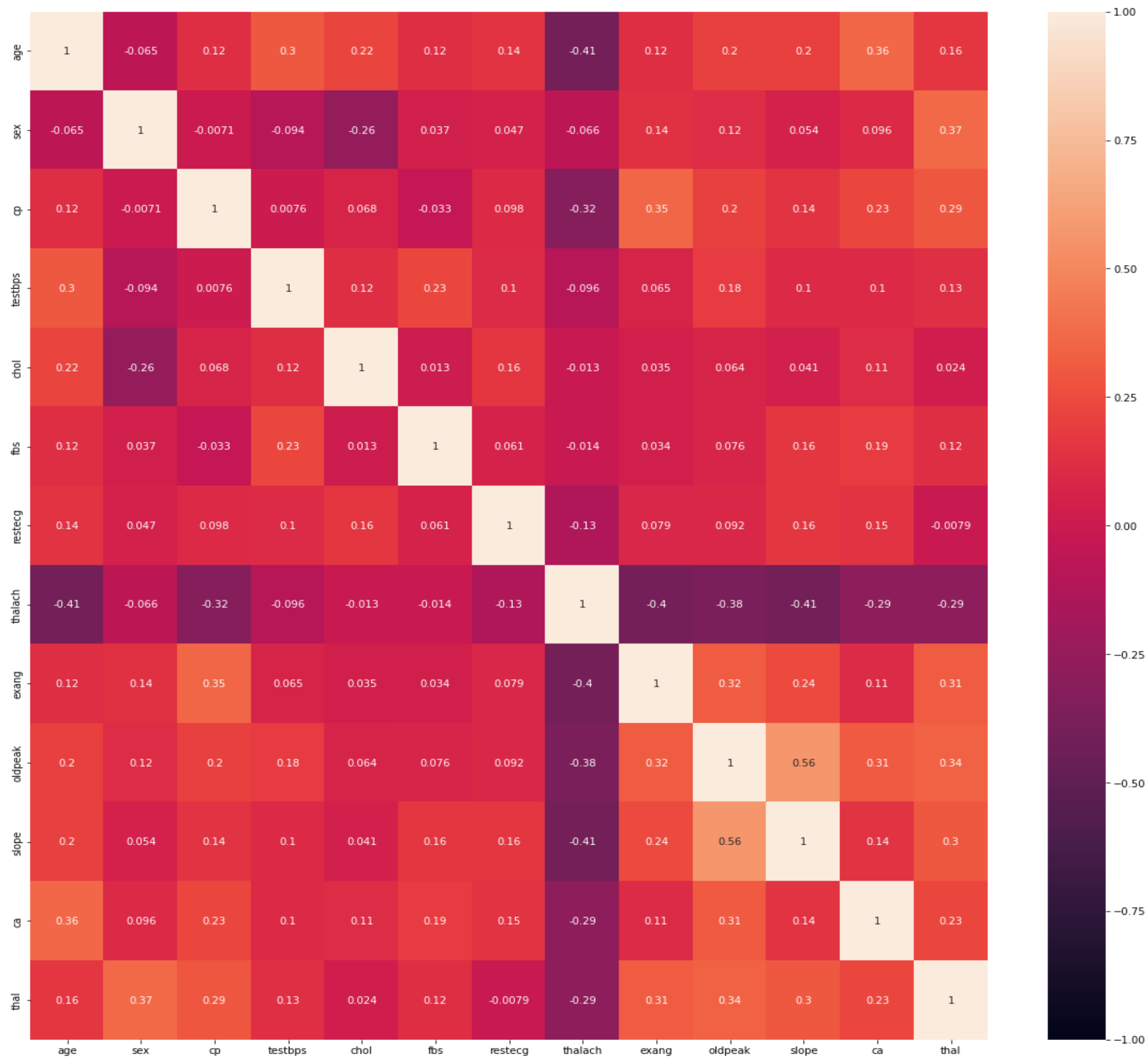
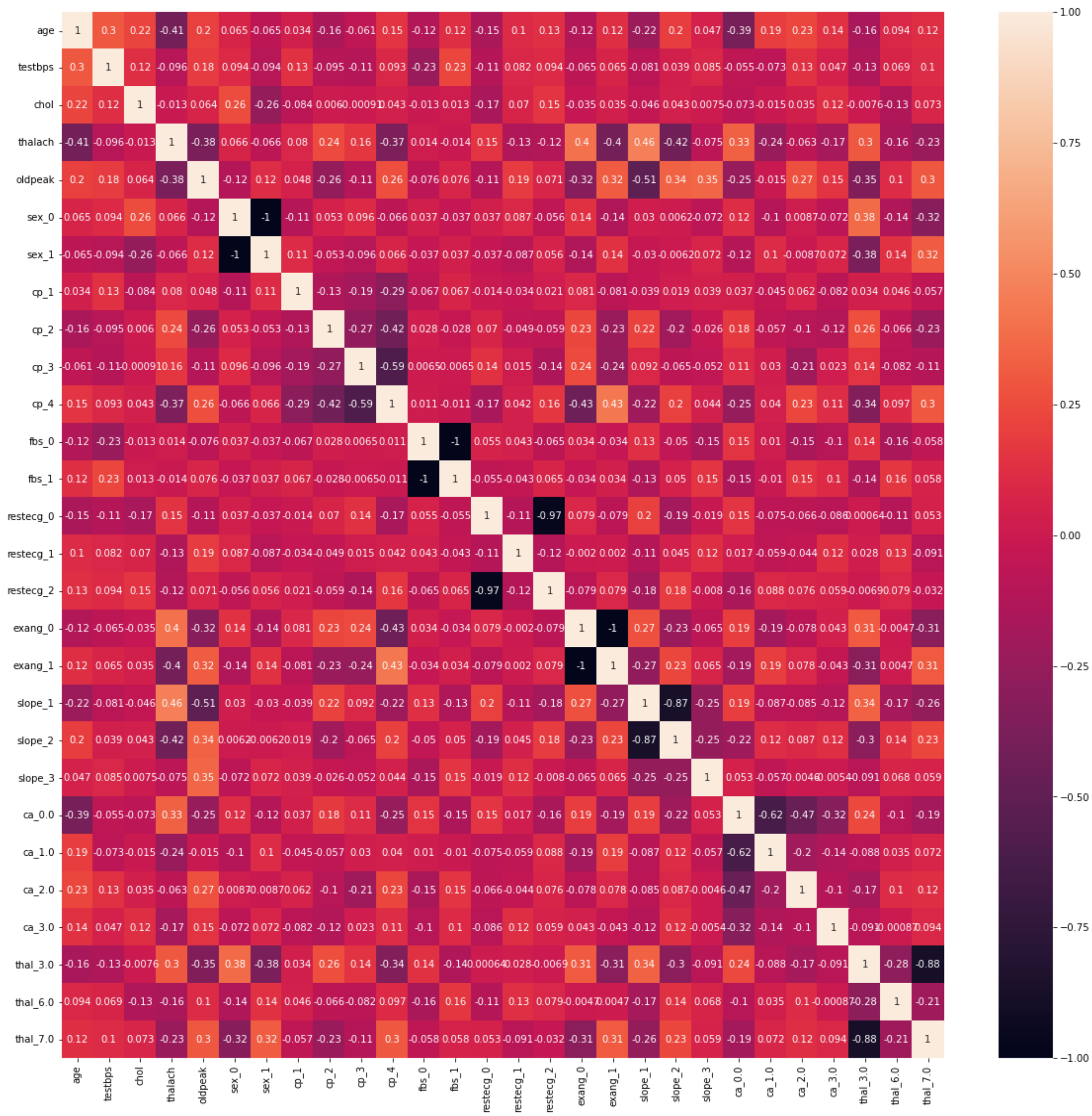


Figure 3.9: Correlation Matrix for Encoded Dataset



Regression Analysis:

A. Multiple Linear Regression

Results for the implementation of multiple linear regression model is examined in this section and since the task is multi-class classification it is expected that this model would perform poorly. Only 54% r-squared score is achieved with an MSE of 0.56 for the dataset that is encoded using the pd.get_dummies function. This was expected since the problem is not regression.

Figure 3.10: Results for Encoding with pd.get_dummies

```
the model performance
------------------------------
MSE is 0.5591298133532832
R-squared score is 0.54
```

Figure 3.11: Results for Not Encoding

```
the model performance
------------------------------
MSE is 0.5737656565566953
R-squared score is 0.52
```

B. Multi-class Classification using Logistic Regression

An R squared score of 0.67 is achieved with the encoded dataset. The performance is improved but an MSE of 0.82 is also observed. This could be due to the inconsistency in the distribution of instances as described in the sections above. The confusion matrix also shows that more samples under the 1 to 4 categories were misclassified. The results for running the learning algorithm with the dataset that is not encoded shows an even worse MSE of 0.95 with an R squared score of 0.68.

Figure 3.12: Results for Encoding with pd.get_dummies

```
the model performance
------------------------------
MSE is 0.8166666666666667
R-squared score is 0.67
```

Figure 3.13: Confusion Matrix

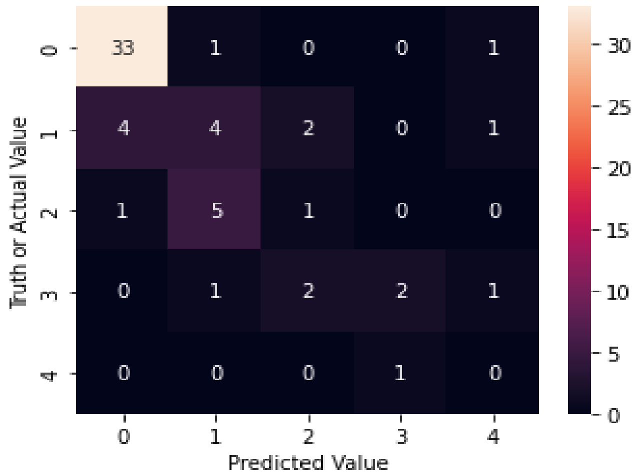
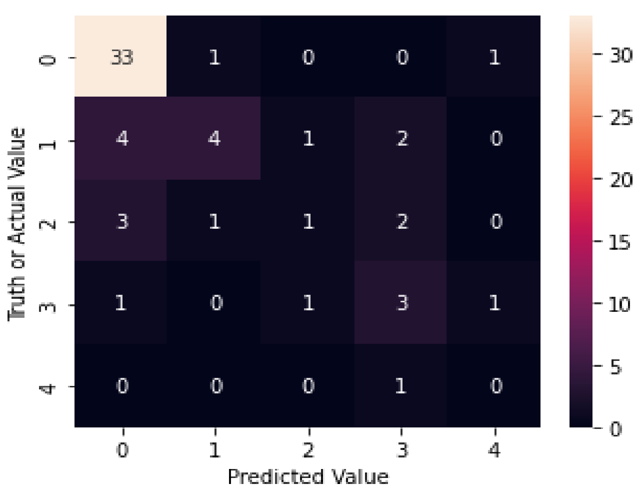


Figure 3.14: Results for Not Encoding

```
[151] print('the model performance')
      print('-----------------------------')
      print('MSE is {}'.format(mse))
      print('R-squared score is {}'.format(r))
```

```
the model performance
-----------------------------
MSE is 0.95
R-squared score is 0.68
```

Figure 3.15: Confusion Matrix



C. Binary Classification using Logistic Regression

In this section the multi-class classification problem is now converted to a binary classification problem and all the attributes labelled as 1, 2, 3, and 4 are put in one category. Here 0 still means that the heart disease is not present, while 1 means that it is present. The distribution of samples is also normalized as shown in the previous sections and there are 160 samples belonging to the 0 category and 137 belonging to 1 category. The results for the dataset that is encoded produces a better r squared score of 0.92 and an MSE of 0.083. The results for the dataset that is not encoded produces an r squared score of 0.88 with an MSE of 0.12.

Figure 3.16: Results for encoding with pd.get_dummies

```
the model performance
------------------------------
MSE is 0.08333333333333333
R-squared score is 0.92
```

Figure 3.17: Confusion Matrix

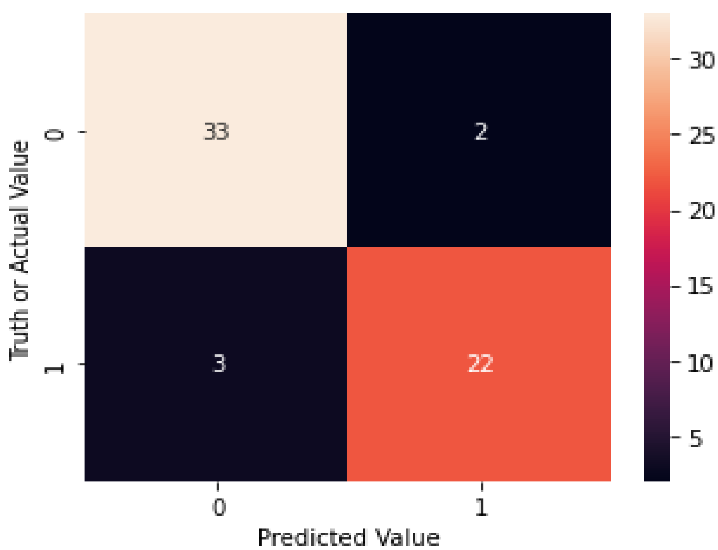
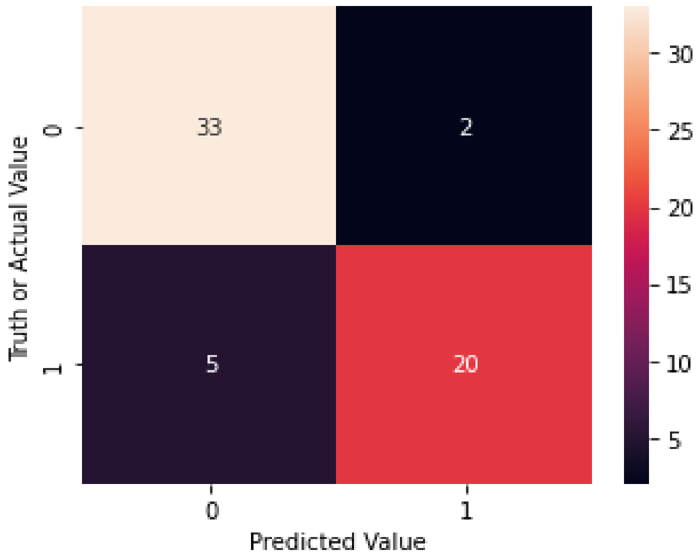


Figure 3.18: Results for Not Encoding

```
the model performance
------------------------------
MSE is 0.11666666666666667
R-squared score is 0.88
```

Figure 3.19: Confusion Matrix



Observations:

Overall, it can be concluded that when the problem is converted to a binary classification problem the Logistic Regression model performs better with lesser error. The uneven distribution of the dataset is the main reason for the poor performance of the Logistic Regression model implemented for multi-class classification problem. This could be improved if synthetic data can be generated to increase the number of samples belonging to all the 5 categories.

Appendix:

The python code for question number 3 is shown in this section. For different cases the variables are adjusted.

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

    import matplotlib
    from matplotlib import pyplot as plt
    import seaborn as sns
    from sklearn.preprocessing import StandardScaler
```

```
[2] from google.colab import drive
    drive.mount('/content/drive/')

    Mounted at /content/drive/
```

```
#load dataset
path='/content/drive/MyDrive/ds_heart/heart_disease_binary.csv'
df=pd.read_csv(path)

df.head()
```

	age	sex	cp	testbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	1	145	233	1	2	150	0	2.3	3	0.0	6.0	0
1	67	1	4	160	286	0	2	108	1	1.5	2	3.0	3.0	1
2	67	1	4	120	229	0	2	129	1	2.6	2	2.0	7.0	1
3	37	1	3	130	250	0	0	187	0	3.5	3	0.0	3.0	0

```
[4] #drop samples with missing values
    df.isnull().sum()

    age         0
    sex         0
    cp          0
    testbps     0
    chol        0
    fbs         0
    restecg     0
    thalach     0
    exang       0
    oldpeak     0
    slope       0
    ca          4
    thal        2
    target      0
    dtype: int64
```

```
[5] df.shape

    (303, 14)
```

```
[6] data=df.dropna()
```

```
data.head()
```

	age	sex	cp	testbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	1	145	233	1	2	150	0	2.3	3	0.0	6.0	0
1	67	1	4	160	286	0	2	108	1	1.5	2	3.0	3.0	1
2	67	1	4	120	229	0	2	129	1	2.6	2	2.0	7.0	1

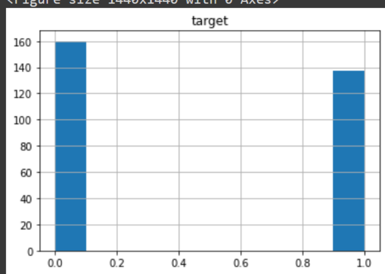
0s completed at 10:02 PM

```
[8] data.shape
```

```
(297, 14)
```

```
[9] plt.figure(figsize=(20,20))
    data.hist('target')
    plt.show('target')
```

```
<Figure size 1440x1440 with 0 Axes>
```




```
[ ] data['target'].value_counts()
```

```
0    160
1    137
Name: target, dtype: int64
```

```
data.describe()
```

```
[ ] #feature selection
    def correlation(dataset, threshold):
        col_corr = set()
        corr_matrix = dataset.corr()
        for i in range(len(corr_matrix.columns)):
            for j in range(i):
                if abs(corr_matrix.iloc[i, j]) > threshold:
                    colname = corr_matrix.columns[i]
                    col_corr.add(colname)
        return col_corr
```

```
[ ] corr_features = correlation(x_train, 0.3)
    len(set(corr_features))
```

```
6
```

```
[ ] corr_features
```

```
{'ca', 'exang', 'oldpeak', 'slope', 'thal', 'thalach'}
```

```
[ ] x_train.drop(corr_features,axis=1)
    x_test.drop(corr_features,axis=1)
```

	age	sex	cp	testbps	chol	fbs	restecg
56	-0.502750	1	3	0.468418	-0.276443	0	0
118	0.936181	1	4	-0.095506	1.592176	1	2
133	-0.392063	1	4	0.468418	0.262952	0	2
24	0.604120	1	4	-0.095506	-0.796575	0	2
265	-1.388246	1	4	0.242848	1.303215	0	0

✓ 0s completed at 10:02 PM

	age	sex	cp	testbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	ta
count	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.00
mean	54.542088	0.676768	3.158249	131.693603	247.350168	0.144781	0.996633	149.599327	0.326599	1.055556	1.602694	0.676768	4.730640	0.46
std	9.049736	0.468500	0.964859	17.762806	51.997583	0.352474	0.994914	22.941562	0.469761	1.166123	0.618187	0.938965	1.938629	0.49
min	29.000000	0.000000	1.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	1.000000	0.000000	3.000000	0.00
25%	48.000000	0.000000	3.000000	120.000000	211.000000	0.000000	0.000000	133.000000	0.000000	0.000000	1.000000	0.000000	3.000000	0.00
50%	56.000000	1.000000	3.000000	130.000000	243.000000	0.000000	1.000000	153.000000	0.000000	0.800000	2.000000	0.000000	3.000000	0.00
75%	61.000000	1.000000	4.000000	140.000000	276.000000	0.000000	2.000000	166.000000	1.000000	1.600000	2.000000	1.000000	7.000000	1.00
max	77.000000	1.000000	4.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	3.000000	3.000000	7.000000	1.00

```
[ ] from sklearn import preprocessing

    x=data.drop('target', axis=1)
    y=data['target']
```

```
[ ] #get dummy variables
    #x=pd.get_dummies(x, columns=['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal'])
    #scale
    standardScaler = StandardScaler()
    columns_to_scale = ['age', 'testbps', 'chol', 'thalach', 'oldpeak']
    x[columns_to_scale] = standardScaler.fit_transform(x[columns_to_scale])
```

```
[ ] x.head()
```

	age	sex	cp	testbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	0.936181	1	1	0.750380	-0.276443	1	2	0.017494	0	1.068965	3	0.0	6.0
1	1.378929	1	4	1.596266	0.744555	0	2	-1.816334	1	0.381773	2	3.0	3.0
2	1.378929	1	4	-0.659431	-0.353500	0	2	-0.899420	1	1.326662	2	2.0	7.0
3	-1.941680	1	3	-0.095506	0.051047	0	0	1.633010	0	2.099753	3	0.0	3.0
4	-1.498933	0	2	-0.095506	-0.835103	0	2	0.978071	0	0.295874	1	0.0	3.0

```
[ ] x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=5)
    x_train.shape,x_test.shape

    ((237, 13), (60, 13))
```

```
[ ] y_train

    267    1
    97     1
    184    1
    170    1
    154    1
          ..
    8      1
    73     1
    119    1
    191    1
    209    1
    Name: target, Length: 237, dtype: int64
```

```
##plot the correlation matrix##
plt.figure(figsize=(20,20))
cor = x_train.corr()
sns.heatmap(cor, vmin=-1,vmax=1, annot=True)
plt.show()
```

Logistic regression:

```
[ ] from sklearn import linear_model
    model=linear_model.LogisticRegression(max_iter=10000)
    model.fit(x_train,y_train)

    LogisticRegression(max_iter=10000)

[ ] y_pred=model.predict(x_test)

[ ] mse=mean_squared_error(y_test,y_pred)

[ ] r=round(model.score(x_test,y_test),2)

[ ] print('the model performance')
    print('-----------------------------')
    print('MSE is {}'.format(mse))
    print('R-squared score is {}'.format(r))

    the model performance
    -----------------------------
    MSE is 0.11666666666666667
    R-squared score is 0.88

[ ] model.score(x_test,y_test)

    0.8833333333333333

[ ] predicted_output = model.predict(x_test)
    predicted_output

    array([1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1,
           1. 0. 1. 0. 0. 0. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.
```

✓ 0s completed at 10:02 PM

Linear regression:

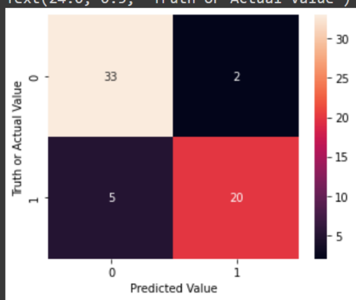
```
] from sklearn import linear_model
  model=linear_model.LinearRegression()
  model.fit(x_train,y_train)

  LinearRegression()

] y_pred=model.predict(x_test)

] mse=mean_squared_error(y_test,y_pred)

] r=round(model.score(x_test,y_test),2)

] print('the model performance')
  print('-----------------------------')
  print('MSE is {}'.format(mse))
  print('R-squared score is {}'.format(r))

  the model performance
  -----------------------------
  MSE is 0.5737656565566953
  R-squared score is 0.52
```

```
[ ] from sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test, predicted_output)
    cm
```

```
array([[33,  2],
       [ 5, 20]])
```

```
[ ] import seaborn as sn
    from matplotlib import pyplot as plt
    plt.figure(figsize = (5,4))
    sn.heatmap(cm, annot=True)
    plt.xlabel('Predicted Value')
    plt.ylabel('Truth or Actual Value')
```

```
Text(24.0, 0.5, 'Truth or Actual Value')
```



References:

[1] *UCI Machine Learning Repository: Heart disease data set*. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Heart+Disease. [Accessed: 13-Feb-2022].

[2] *UCI Machine Learning Repository: Heart disease data set*. [Online]. Available: http://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data. [Accessed: 13-Feb-2022].