School of Computer Science Engineering and Technology
Assignment-03

Course- B.Tech	**Type-** Core
Code-23CS106	**Course Name-** Artificial Intelligence & Machine Learning
Year- 2024-2025	**Semester-** Even, **Instructor:** Prof. E.L.N. Kiran
Date– 23-01-2024	**Batch-** AIML-A,B

1 Implement Linear Regression Model Using US Housing Data

Part 1 – Import the required Python, Pandas, Matplotlib, Seaborn packages

1. Load the US Housing data into a dataframe using `pandas`
2. Check the data types of each feature(column) in the dataset.
3. Generate a summary of the dataset for `min, max, stddev, quartile vales for 25%,50%,75%,90%,`
4. List the names of columns/features in the dataset
5. Generate a pairplot of the features of the dataset.
6. Generate a correlation matrix and heatmap for the features
7. Create a list of dependent variable to independent variables to understand regression among the features. From the data include `Price` to other numerical variables of the Housing data.

Part 2 – Model training and Fit the data to Model

1. Split the data generated from list created as `X, Y` is distributed using *train_test_split* function as $X\_train, Y\_train, X\_test, Y\_test$
2. Apply the linear regression model of `sklearn` package
3. Fit the data to the Linear Model using `fit`
4. Check the intercepts and slope for the data and compute the `cumulative distribution function(cdf)`

Part 3 – Model Evaluation Metrics

1. Calculate the standard error and t-statistic for the coefficients.
2. Sort all the coefficients based on the cdf. Generate the scatter plots for the other features considering price as dependent variable.
3. Compute the R^2 for the coeffiecients using $metrics.r2\_score()$

4. Plot the predictions of Linear Regression Model - histogram, scatterplot
5. Generate the evaluation regression error metrics - `MAE, SSE, RMSE` ,R^2 `using metrics`

2 Compute the MinMax value between Observed Price and Expected Price for the US Housing Data

1. Write the python code to compute MinMax value of a Feature within Housing data. We compute the MinMax value using the equation.

$$L\_minmax = \frac{L\_minmax - min(L\_minmax)}{max(L\_minmax) - min(L\_minmax)}$$

2. Normalize the data and Print the MinMax value, plot the distribution of feature.

Part 1.1– Solution

```
#Import the python packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

#Read the data
df = pd.read_csv("USA_Housing.csv")
df.head()

#Check the column datatype, quartiles, names of the features
df.info(verbose=True)
df.describe(percentiles=[0.1,0.25,0.5,0.75,0.9])
df.columns

#Visualize the data using pairplots,
#correlation heatmap between price and other features
sns.pairplot(df)
df['Price'].plot.hist(bins=25,figsize=(8,4))
df['Price'].plot.density()
df.corr()
plt.figure(figsize=(10,7))
sns.heatmap(df.corr(),annot=True,linewidths=2)
```

```
#Generate a data frame list of all features,
#Put numerical features to X and Price feature to Y to compute regression
l_column = list(df.columns) # Making a list out of column names
len_feature = len(l_column) # Length of column vector list
l_column

X = df[l_column[0:len_feature-2]]
y = df[l_column[len_feature-2]]
print("Feature set size:",X.shape)
print("Variable set size:",y.shape)
x.head(), y.head()

#Split the data intro Train and Test data sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=123)
print("Training feature set size:",X_train.shape)
print("Test feature set size:",X_test.shape)
```

```
print("Training variable set size:",y_train.shape)
print("Test variable set size:",y_test.shape)
```

Part 1.2 – Solution

```
# Import the sklearn linear regression model
from sklearn.linear_model import LinearRegression
from sklearn import metrics

# Creating a Linear Regression object 'lm'
m = LinearRegression()

# Fit the model on to the 'lm' object itself. Find the intercept, slope value
lm.fit(X_train,y_train)
print("The intercept term of the linear model:", lm.intercept_)
print("The coefficients of the linear model:", lm.coef_)

#Compute cumulative distribution for the feature (cdf)
#idict = {'Coefficients':lm.intercept_}
#idf = pd.DataFrame(data=idict,index=['Intercept'])
cdf = pd.DataFrame(data=lm.coef_, index=X_train.columns, columns=["Coefficients"])
#cdf=pd.concat([idf,cdf], axis=0)
cdf
```

Part 1.3 – Solution

```
#Compute the std.error and t-statistic for features
n=X_train.shape[0]
k=X_train.shape[1]
dfN = n-k
train_pred=lm.predict(X_train)
train_error = np.square(train_pred - y_train)
sum_error=np.sum(train_error)
se=[0,0,0,0,0]
for i in range(k):
r = (sum_error/dfN)
r = r/np.sum(np.square(X_train[list(X_train.columns)[i]]-
    X_train[list(X_train.columns)[i]].mean()))
se[i]=np.sqrt(r)
cdf['Standard Error']=se
cdf['t-statistic']=cdf['Coefficients']/cdf['Standard Error']
```

```
cdf

#print the sorted features based on error metrics
print("Therefore, features arranged in the order of importance for
          predicting the house price\n",'-'*90,sep='')
l=list(cdf.sort_values('t-statistic',ascending=False).index)
print('>\n'.join(l))

#Generate a scatter plot for all error metric using Price as feature
l=list(cdf.index)
from matplotlib import gridspec
fig = plt.figure(figsize=(18, 10))
gs = gridspec.GridSpec(2,3)
#f, ax = plt.subplots(nrows=1,ncols=len(l), sharey=True)
ax0 = plt.subplot(gs[0])
ax0.scatter(df[l[0]],df['Price'])
ax0.set_title(l[0]+" vs. Price", fontdict={'fontsize':20})

ax1 = plt.subplot(gs[1])
ax1.scatter(df[l[1]],df['Price'])
ax1.set_title(l[1]+" vs. Price",fontdict={'fontsize':20})

ax2 = plt.subplot(gs[2])
ax2.scatter(df[l[2]],df['Price'])
ax2.set_title(l[2]+" vs. Price",fontdict={'fontsize':20})

ax3 = plt.subplot(gs[3])
ax3.scatter(df[l[3]],df['Price'])
ax3.set_title(l[3]+" vs. Price",fontdict={'fontsize':20})

ax4 = plt.subplot(gs[4])
ax4.scatter(df[l[4]],df['Price'])
ax4.set_title(l[4]+" vs. Price",fontdict={'fontsize':20})

#R-squared error metric
print("R-squared value of this fit:",round(metrics.r2_score(y_train,train_pred),3))

#Predict error metrics, plot the regression evaluation metrics
#- MAE, SSE, RMSE, Rsq error
predictions = lm.predict(X_test)
print ("Type of the predicted object:", type(predictions))
print ("Size of the predicted object:", predictions.shape)

plt.figure(figsize=(10,7))
```

```python
plt.title("Actual vs. predicted house prices",fontsize=25)
plt.xlabel("Actual test set house prices",fontsize=18)
plt.ylabel("Predicted house prices", fontsize=18)
plt.scatter(x=y_test,y=predictions)

plt.figure(figsize=(10,7))
plt.title("Histogram of residuals to check for normality",fontsize=25)
plt.xlabel("Residuals",fontsize=18)
plt.ylabel("Kernel density", fontsize=18)
sns.histplot([y_test-predictions])

plt.figure(figsize=(10,7))
plt.title("Residuals vs. predicted values plot (Homoscedasticity)\n",fontsize=25)
plt.xlabel("Predicted house prices",fontsize=18)
plt.ylabel("Residuals", fontsize=18)
plt.scatter(x=predictions,y=y_test-predictions)

print("Mean absolute error (MAE):", metrics.mean_absolute_error(y_test,predictions))
print("Mean square error (MSE):", metrics.mean_squared_error(y_test,predictions))
print("Root mean (RMSE):", np.sqrt(metrics.mean_squared_error(y_test,predictions)))

print("R-squared value of predictions:",round(metrics.r2_score(y_test,predictions),3))
```