Faculty of computers and artificial intelligence

**Cover sheet**

**AI330 Machine Learning Project**

**Team no. :**

|  |  |
| --- | --- |
| Name | ID |
| محمد ايمن عبدالفتاح | 20210750 |
| عبدالله خالد عبدالله | 20210543 |
| محمد احمد ابراهيم | 20210724 |
| امنيه خليل محمد | 20210181 |
| اميره اشرف عبدالحميد | 20210184 |
| الاء ياسر محمدي | 20210174 |

# Numerical dataset

------------------------------------------------------

import libraries :

import pandas : For data manipulation and analysis. import numpy : For numerical operations.

from sklearn.impute

import SimpleImputer :used to replace missing values in the dataset with a chosen strategy .

from sklearn.preprocessing import StandardScaler, OneHotEncoder

-StandardScaler : scales the features to have a mean of 0 and standard deviation of 1.

-OneHotEncoder : used for converting categorical variables into binary vectors.

from sklearn.model\_selection

import train\_test\_split : is used to split the dataset into training and testing sets.

from sklearn.linear\_model import LinearRegression: is a linear regression model.

from sklearn.neighbors

import KNeighborsRegressor : is a regression model based on k-nearest neighbors.

General information about dataset

|  |  |
| --- | --- |
| Name  housing |  |
| Total no. of samples  19353 rows × 10 columns |  |
| No. of samples in training\validation | 15482 rows × 13 columns |
| No. of samples in testing  3871 rows × 13 columns |  |

-pd.read\_csv: function from the Pandas library take the path of dataset to read a file and store its contents in a DataFrame .

-housing.head(5): Displays the first 5 rows of the DataFrame

load the data and display :

--------------------------------------------------------

Count the occurrences :

-def getOutliers(dataframe, column) :

This function helps in identifying and

removing outliers from a specified column in a DataFrame

-dataframe: The input containing the data.

-column:The name of the column for which outliers will be identified.

-IQR: Interquartile Range is calculated to identify the range within which data is considered normal.

-lower and upperBound: Lower and upper bounds for identifying outliers.

-outliers: DataFrame containing rows with outliers.

-outlierRemoved: DataFrame with outliers removed.

-----------------------------------------------------------------

Identify and remove outliers :

-value\_counts():

this method on the 'ocean\_proximity' column of DataFrame to count the occurrences of each value in that column.

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error: are metrics for evaluating the performance of regression models.

from sklearn.metrics

import accuracy\_score : accuracy\_score is a metric for classification models.

handling missing values :

------------------------------------------------------------------------------

-housing.isnull().sum():

Check for any remaining missing values in the entire DataFrame 'housing' and print the count of missing values for each column.

-housing.iloc[:, 4:5].head(): Display the first few rows of the column at index

4 in the DataFrame 'housing'. This is done using the .iloc indexer for selection.

-imputer = SimpleImputer(strategy="median"):

Create a SimpleImputer object named imputer with the imputation strategy set to "median"

This means missing values in the selected column will be replaced with the median value of that column.

-housing.iloc[:, 4:5] = imputer.fit\_transform(housing.iloc[:, 4:5]):

Fit the imputer on the selected column and transform it

by replacing missing values with the computed median. The result is then assigned back to the original DataFrame.

- housing.drop("median\_house\_value",axis=1) :

drops the rows median\_house\_value and assigns the data to the variable X.

- X.head() :

Displays the first few values of the X variable, showing all dataframe without "median\_house\_value" column.

- y = housing["median\_house\_value"]:

Extracts the "median\_house\_value" column from the DataFrame and assigns it to the variable y.

- y.head():

Displays the first few values of the 'y' variable, showing the extracted "median\_house\_value" column.

- get\_dummies (X['ocean\_proximity']) :

Creates dummy variabls for column 'ocean\_proximity'

to converts categorical values into numerical representations using one-hot encoding.

Each unique category in the original column gets transformed into a new binary

column, and the presence of the category is indicated by 1 or 0, respectively.

- train\_test\_split() :

Split the dataset into 80% train and 20% test dataset.

- fit(): Trains the linear regression model using the training data.

- predict(): Makes predictions on the test set. Performance Evaluation:

- R-squared (): A measure of how well the predicted values match the actual values.

- MAE (mean\_absolute\_error):

average absolute differences between predicted and actual values.

- MSE (mean\_squared\_error):

average of squared differences between predicted and actual values.

(1) - Linear regression madel :

---------------------------------------------------------------------

- scaler.fit\_transform():

standardize training and test dataset

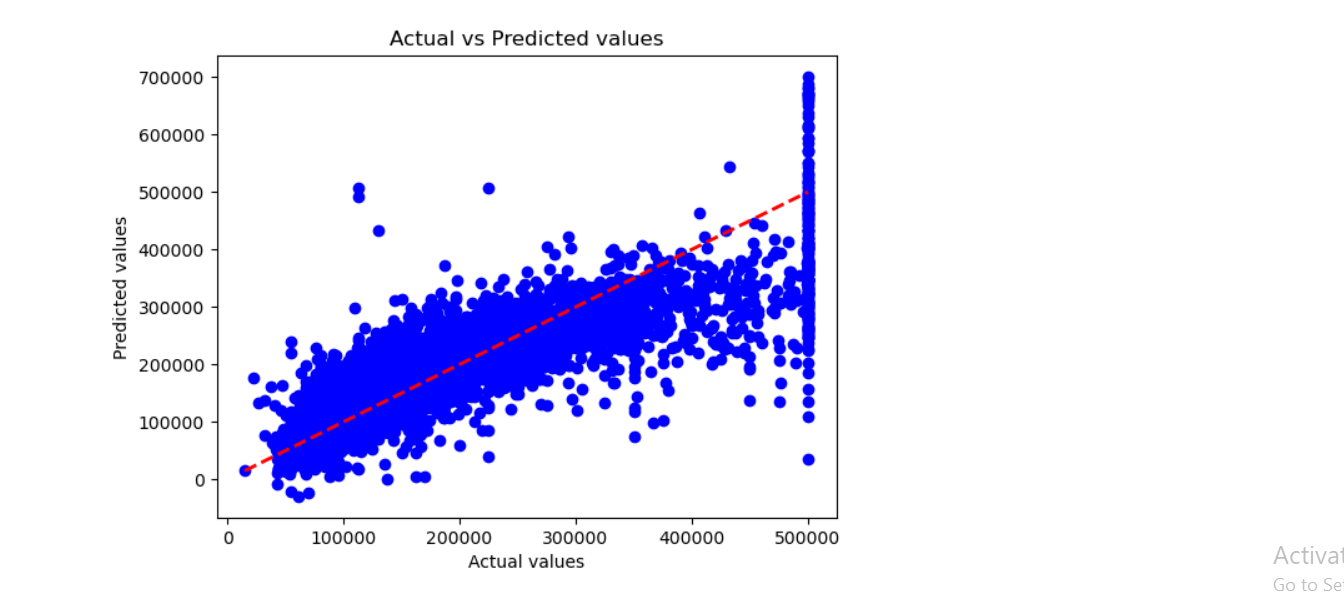
- concat():

Concatenate the encoded categorical features with the numerical features

**Graphical Representation:**

**- Scatter plot of Actual vs Predicted values.**

**- Regression line plotted in red.**

****

(2) - knn Regression model:

KNeighborsRegressor :

-n\_neighbors=5 :

Specifies the number of neighbors to consider.

-weights='uniform' :

Specifies that all neighbors have equal weight.

-algorithm='auto' :

Automatically selects the best algorithm based on the input data.

- Fit the model :

this method to fit the model to training data (X\_train , y\_train).

- Make predictions :

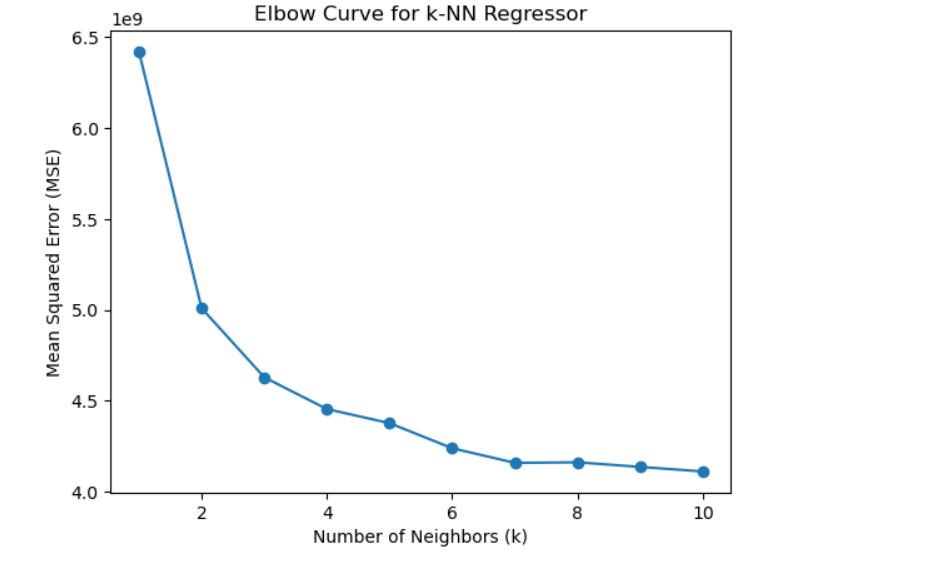
used the trained model to predict testingset (X\_test).

The score method gives the coefficient of determination R^2 of the prediction.

- Evaluate the model:

calculat and print the Mean Absolute Error (MAE) and Mean Squared Error (MSE) as evaluation metrics for model.

Elbow Curve for k-NN Regressor

******

# Image dataset

---------------------------------------------------------------------------------------------

load image data:

load\_data():

retrieve image directories from the specified data\_directory load images and labels, limiting the number of directories to num\_directories

return a tuple of images and labels

preprocess\_images\_for\_hog():

Resize and convert images to grayscale for HOG feature extraction

Return processed images as a NumPy array

- test /train\_hog\_features:

Represents an empty list to store (HOG) features for the test/train images.

- test/train\_images\_processed: Initiates a loop to iterate through each preprocessed test/train image in the test\_images\_processed list.

train\_hog\_features ,test\_hog\_features :compute HOG features

-------------------------------------------------------------------------------------

General information about dataset

|  |  |
| --- | --- |
| Name  traffic sign |  |
| No. of classes  5 / 324  and features |  |
| Total no. of samples  20000image |  |
| Size of image  32\*32 |  |
| No. of samples in training\validation  train\_image 7860 |  |
| No. of samples in testing  test\_images 4440 |  |

- hog\_img = hog(img, visualize=False): Computes the HOG features for the current test/train image (img) using the hog function.

- test\_hog\_features.append(hog\_img):

Appends the computed HOG features for the current test image to the test\_hog\_features list.

---------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------

(3) - LogisticRegressionModel:

- penalty='l2': the regularization term used in the logistic regression model.

- solver='sag': Specifies the optimization algorithm used to train the logistic regression model.

- C=1.0: Inverse of regularization strength.

- max\_iter=1000: Maximum number of iterations for optimization during training.

- fit(): Train the logistic regression model using the HOG features of training images and corresponding labels

- LogisticRegressionModel.score():

Calculate and display the training score (accuracy) of the logistic regression model on the training set.

- LogisticRegressionModel.classes\_:

Display the unique classes in the training data

- LogisticRegressionModel.n\_iter\_:

Display the actual number of iterations taken by the optimization algorithm during training.

- predict():

Invokes the predict method of the trained Logistic Regression model to obtain the predicted class

labels for the test set based on the HOG features (test\_hog\_features).

- y\_pred\_prob: Contains the predicted class probabilities for each class in the test set using the trained Logistic Regression model.

Each row corresponds to a test sample, and each

column represents the predicted probability for a specific class.

- predict\_proba(test\_hog\_features):

Invokes the predict\_proba method of the trained Logistic Regression model to obtain the predicted class probabilities for the test set based on the HOG features.

- accuracy\_score():

Uses the accuracy\_score function from scikit-learn to calculate the accuracy by comparing the true class labels (test\_labels) with the predicted class labels (y\_pred).

- MAEValue:

measures the average absolute differences between the true and predicted values.

- mean\_absolute\_error():

Uses the mean\_absolute\_error function from scikit-learn to calculate the MAE by comparing the true class labels (test\_labels) with the predicted class labels (y\_pred).

- CM: Represents the confusion matrix for the Logistic Regression model on the test set

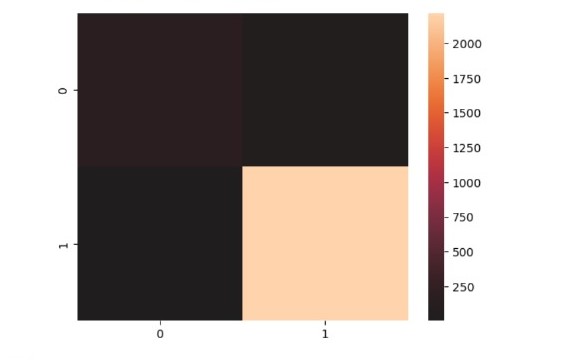
- confusion\_matrix(): Uses the confusion\_matrix function from scikit- learn to calculate the confusion matrix by comparing the true class labels (test\_labels) with the predicted class labels (y\_pred).

- sns.heatmap(CM, center=True):

Uses the heatmap function from the Seaborn library to create a heatmap visualization of the confusion matrix (CM).

- plt.show():

Displays the heatmap plot

******

(4) - kmean model :

- KMeans(n\_clusters=8, init='k-means++', n\_init=10, max\_iter=1000, tol=0.0001, random\_state=42):

Instantiates a KMeans clustering model with the specified hyperparameters, init='k-means++' specifies the initialization method, 10 is the number of times the algorithm will be run with different centroid seeds, max\_iter sets the maximum number of iterations, tol is the tolerance to declare convergence, and random\_state ensures reproducibility.

KMeansModel.fit():

Fits the KMeans model to the training data (HOG\_images\_train), assigning cluster labels to each data point.

- print('KMeansModel Train Score is:,',KMeansModel.score):

Prints the total squared distance of samples to their cluster center, which is the negative of the inertia. It's a measure of how well the clustering has been performed

- print('KMeanscenters are: ' KMeans.cluster\_centers\_): Prints the coordinates of cluster centers.

- print('KMeansModel labels are: ', KMeansModel.labels\_): Prints the assigned cluster labels for each data point.

- print('KMeansModel inertia is: ', KMeansModel.inertia\_):

Prints the sum of squared distances of samples to their closest cluster center.

- print('KMeansModel No. of iteration is: ', KMeansModel.n\_iter\_): Prints the number of iterations

-----------------------------------------------------------------------------------

warnings.filterwarnings("ignore", category=FutureWarning, module=" sklearn.cluster.\_kmeans"):

Ignores FutureWarnings related to KMeans in order to suppress

ilist.append(KMeansModel.inertia\_): Appends the inertia value to the list.

- plt.plot(range(1, n), ilist):

Plots the elbow curve to visualize the relationship between the number of clusters and inertia

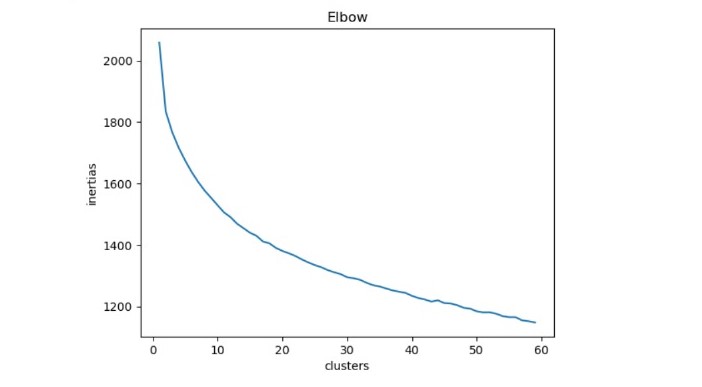
- plt.title('Elbow Method for Optimal Cluster Count'): Sets the title of the plot.

- plt.xlabel('Number of Clusters'): Sets the x-axis label.

- plt.ylabel('Inertia'): Sets the y-axis label.

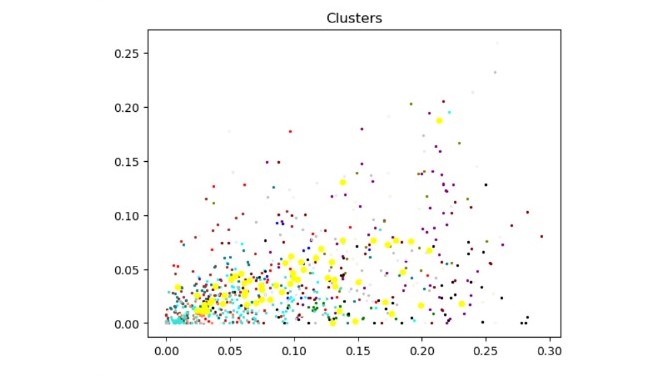
- plt.show():

Displays the elbow curve plot.

******

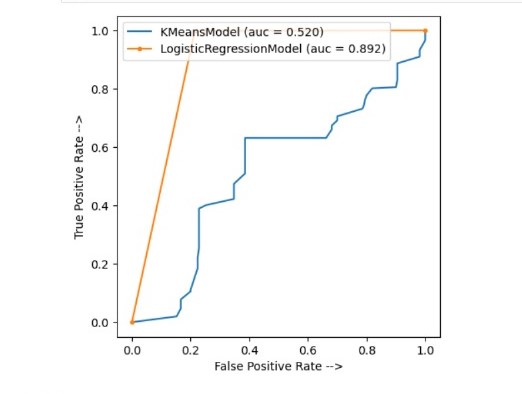
- plt.show():

Displays the cluster plot.

******

- plt.show():

Displays the roc\_curve,auc

****