|  |  |
| --- | --- |
| ***ID*** | ***Name*** |
| ***20210311*** | ***دياب موسى دياب*** |
| ***20210378*** | ***زياد نبيل جابر*** |
| ***20210384*** | ***زياد ياسر محمد*** |
| ***20210368*** | ***زياد عمرو محمد*** |
| ***20210300*** | ***خالد سامح محمد*** |
| ***20210762*** | ***محمد حسن محمد*** |

1. ***Stanford Dogs Dataset***

**Dataset link :**

***Link:*** *<http://vision.stanford.edu/aditya86/ImageNetDogs/main.html>*

**About dataset :**

The Stanford Dogs dataset is a collection of images of various dog breeds, collected and annotated by researchers at Stanford University. It consists of photos of dogs belonging to 120 different breeds, with a total of around 20,000 images.

This dataset has been built using images and annotation from ImageNet for the task of fine-grained image categorization. Contents of this dataset:

**Implementation details :**

We used two algorithms **K-Means** and **Logistic Regression**

**Pre-Processing:**

Extracts features by HOG algorithm for train and test data, and make a resize for the images and Convert to grayscale if it has three channels

And split dataset into 20% test and 80% train by train\_test\_slpit

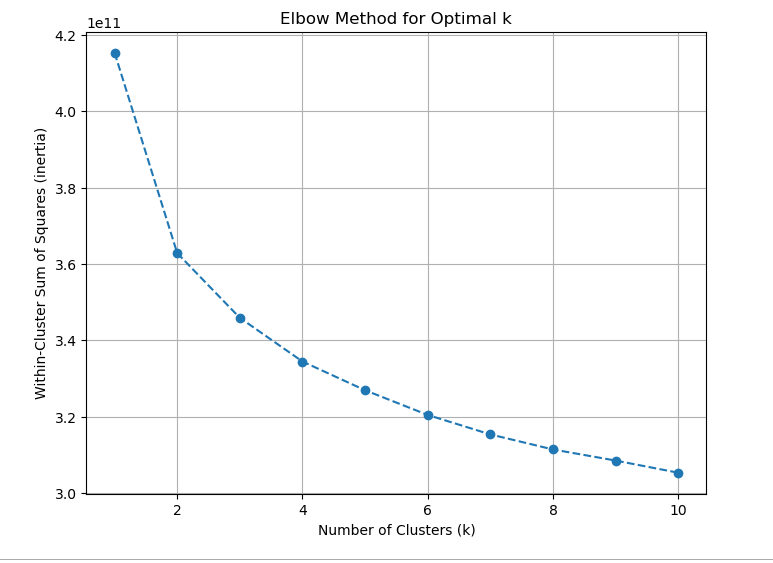
algorithm.

1. **K-Means:**

Goal: Classify Dogs into five breeds of dogs (0,1,2,3,4)

Explain Code: select K randomly and loop from 2 to 11 , in each

iteration save inertia in list after end loop compare all inertias and plot Elbow to select right K(loss curve)



**Result:**

-The Inertia: 415316062447.3274

-Silhouette Score: 0.061699429101101474

-Visualize the cluster:



1. **Logistic Regression:**

**Goal:** Detect breed

**Explain Code**: We don't need to sum image features; we used

dataset with all image features.

**Train Score**: 0.5913272010512484

**Test Score**: 0.3717277486910995

**Classes is**: [0 1 2 3 4]

**Iterations is**: [132]

**Intercept is**: [-1.25949007]

**Confusion Matrix:**

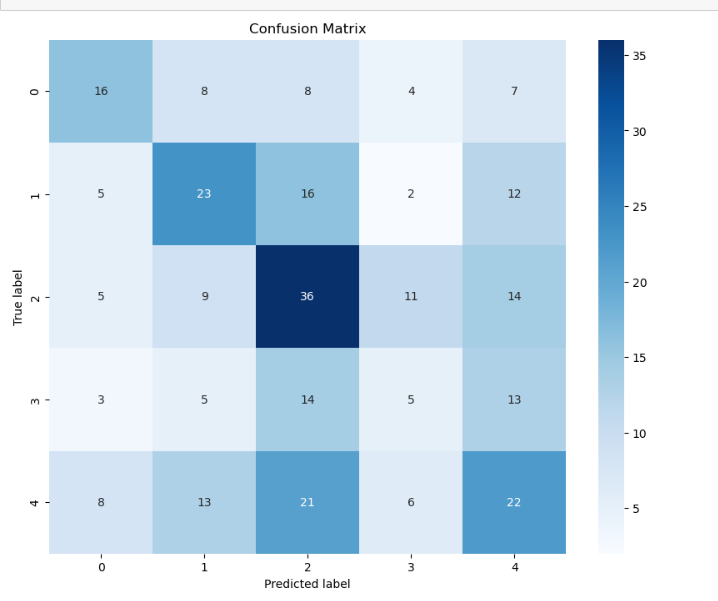
[[16 8 8 4 7]

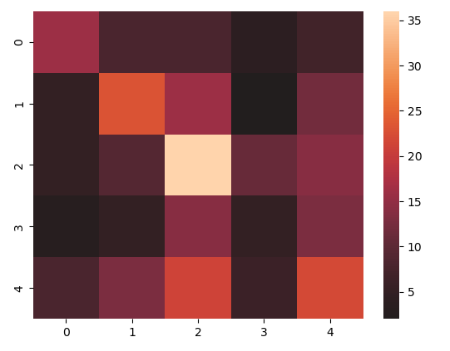
[ 5 23 16 2 12]

[ 5 9 36 11 14]

[ 3 5 14 5 13]

[ 8 13 21 6 22]]





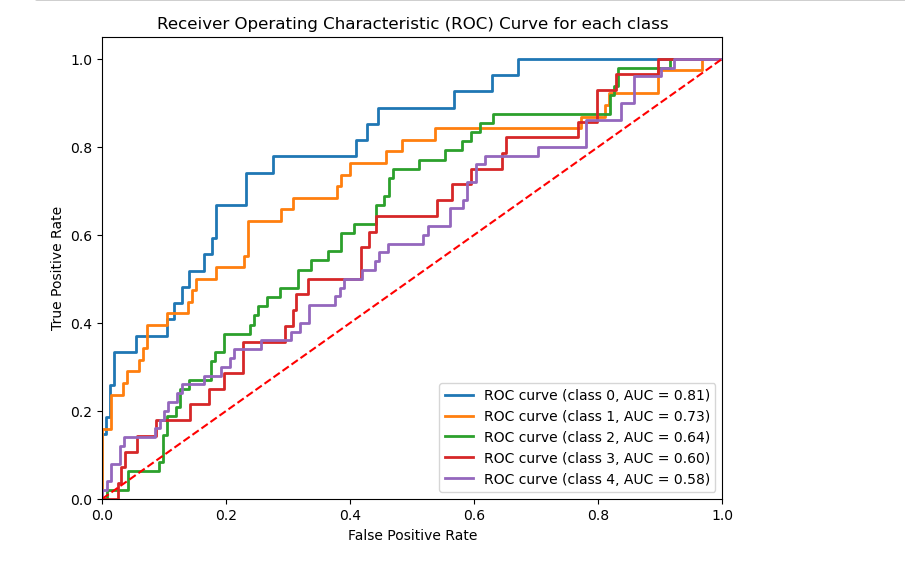
**Accuracy Score**: 0.3717277486910995

**Precision Score is**: 0.3717277486910995

**Precision Recall Score is**: ( (0.3717277486910995, 0.3717277486910995, 0.3717277486910995, None)

**Zero One Loss Value**: 120

**ROC curve**:



1. **California housing prices**

**Dataset link:**

<https://www.kaggle.com/datasets/camnugent/california-housingprices>

**About Dataset:**

**Context**

This is the dataset used in the second chapter of Aurélien Géron's

recent book 'Hands-On Machine learning with Scikit-Learn and

TensorFlow'. It serves as an excellent introduction to

implementing machine learning algorithms because it requires

rudimentary data cleaning, has an easily understandable list of

variables and sits at an optimal size between being to toyish and

too cumbersome.

The data contains information from the 1990 California census. So

although it may not help you with predicting current housing

prices like the Zillow Zestimate dataset, it does provide an

accessible introductory dataset for teaching people about the

basics of machine learning.

**Content**

The data pertains to the houses found in a given California district

and some summary stats about them based on the 1990 census

data. Be warned the data aren't cleaned so there are some

preprocessing steps required! The columns are as follows, their

names are pretty self explanitory:

Longitude

Latitude

housing\_median\_age

total\_rooms

total\_bedrooms

Population

Households

median\_income

median\_house\_value

ocean\_proximity

**Implementation details:**

**Pre-Processing:**

a) Encoding all text into integer by oneHotEncoding algorithm

b) drop all null values

c) Check the correlation to determine which features is important

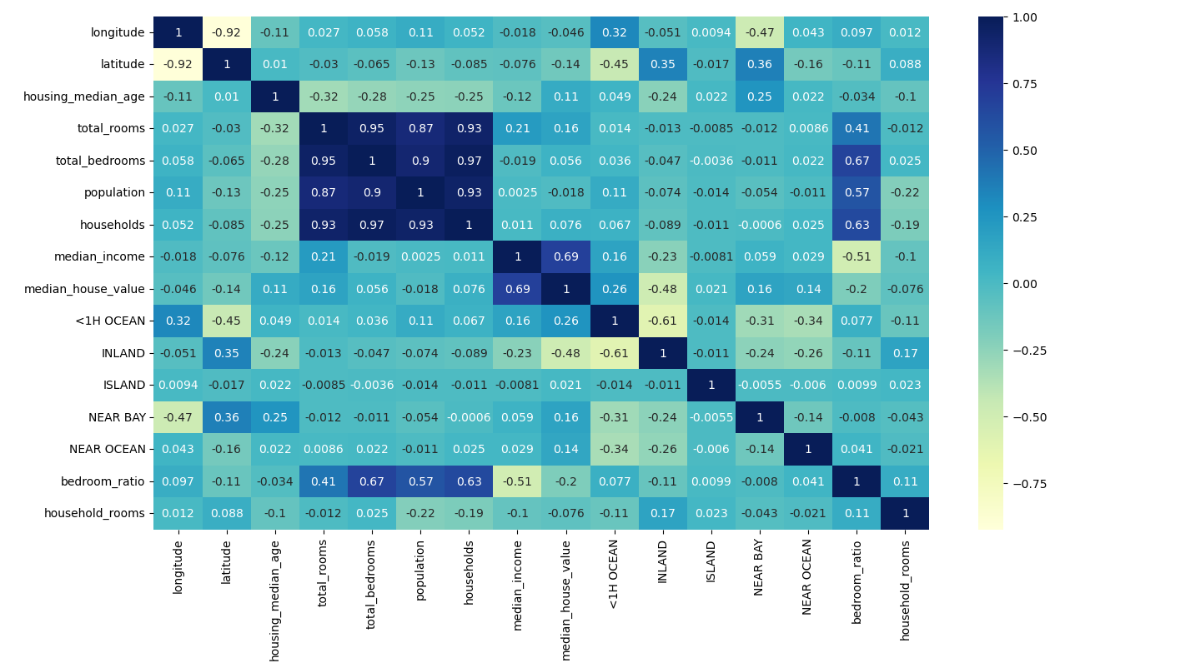
d) scale some features using standard scaler

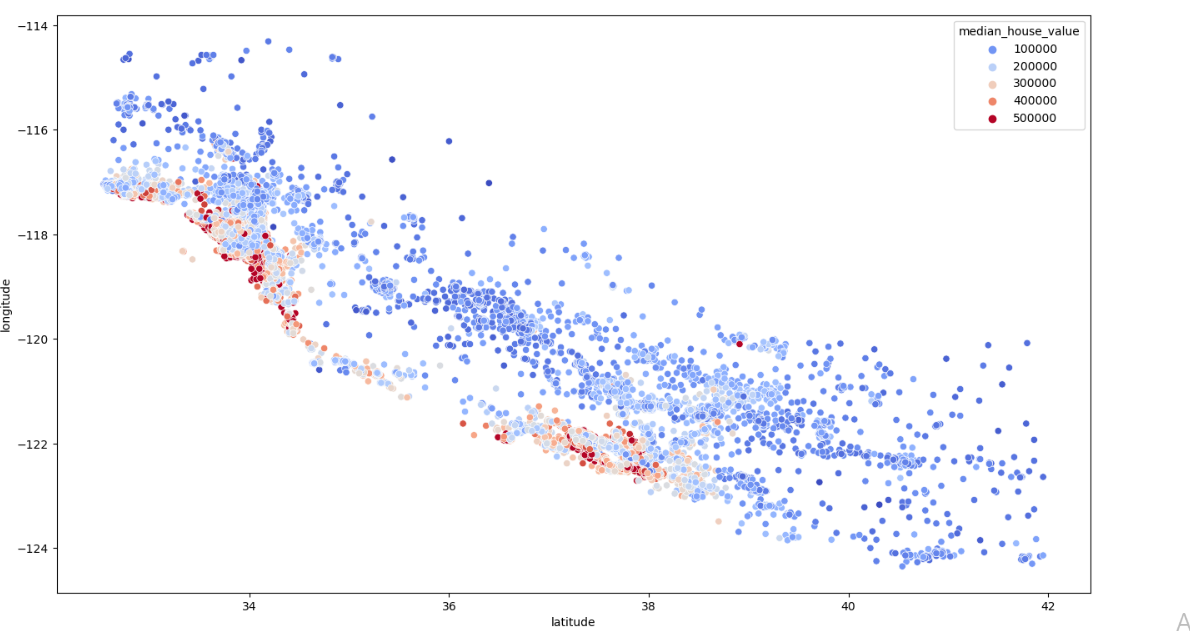
e) create new features

f) Split data into features and target values

g) Split into train and test by train-test-split where test size is 20%

and train size is 80%





**a) Linear Regression**

Goal: predict the price

Code: used linear regression from sklearn.lineae\_model

Results:

Accuracy of linear regression(test): 0.6748273817082482

Accuracy of linear regression(train):0.6701724789136202

- Then we used random forest model to enhance the

accuracy

The accuracy of random forest: 0.8247593087923877

- Then we tried to enhance the accuracy by using

GridSearchCV algorithm, but we got the same

accuracy of random forest.

**B) K Nearest Neighbors (KNN)**

Goal: predict the price

Code: set K to 28

y\_predd[:10]==>array([ 96485.71428571, 68550. , 237239.32142857, 170932.14285714,

64214.28571429, 202817.85714286, 235096.46428571, 113625. ,

186389.32142857, 396543.10714286])

Result:

The accuracy of knn is : 0.6852001171645001