Numerical Dataset

Dataset (House Prices - Advanced Regression Techniques)

- A Dataset that allows as to predict the prices of certain houses based on some feature values.
- It contains two main files (Train, Test), and 80 feature values.
- The Train file contains 1460 row, while the Test file contains 1459 row
- The Dataset has some columns that contains missing values

Handling missing values (In train.csv and test.csv)

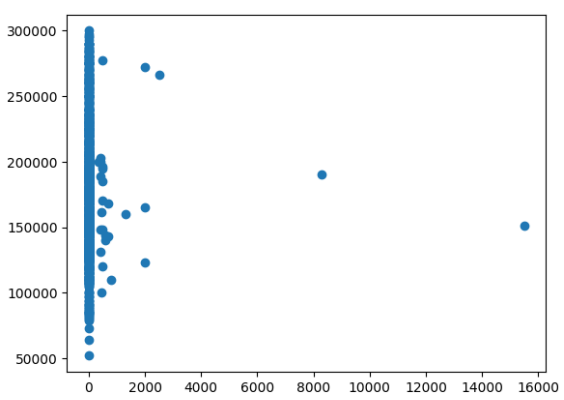
- The columns that contain categorical missing values are handled using mode such as : (FireplaceQu, GarageType, GarageFinish)
- The columns that contain numerical missing values are handled using mean such as : (LotFrontage, GarageYrBlt, MasVnrArea)
- There exists columns that contains so many missing values that will be hard to replace so they are dropped : (Alley, PoolQC, Fence, MiscFeature)

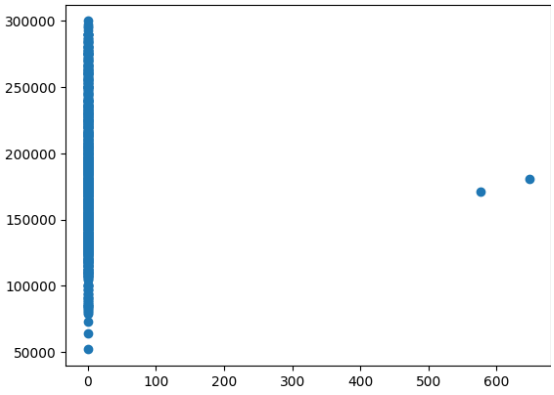
Encoding categorical columns

- Columns that contains categorical values must be encode, so we implemented function : cat_onehot_encoding()
- This function represents categorical data as numerical data in a way that can be used by machine learning algorithms
- It involves converting each unique category in a dataset into a binary vector
- In our code we concatenated the train and test data frames and performed the encoding process in one step
- After that of course there will be duplicated values because of the concatenating process so they were removed
- Then that data frames returns to their initial state (Train, Test)

Handling outliers

- There where some columns that contains outlier values so they are handled
- Those columns are (YearRemodAdd, MasVnrArea, BsmtFinSF2, TotalBsmtSF, LowQualFinSF, GrLivArea, EnclosedPorch, LotFrontage, LotArea, PoolArea, MiscVal)
- Some represented examples:

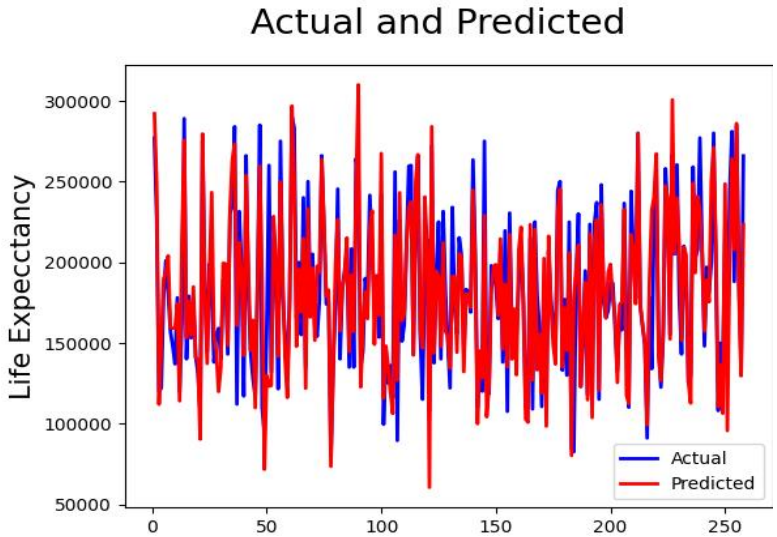




Linear regression training

- We split the dataset into training and testing sets in terms of x and y
- We call the linear regression function from sklearn and pass to it x_train and y_train
- We calculate the different evaluation metrics :
- **Mean Squared Error (MSE) for Linear Regression: 507406784.063794**
- **Mean Absolute Error (MAE) for Linear Regression: 15375.78958075746**
- **R-squared: 0.7998721490296342**

- A graph that shoes Actual VS Predicted values:



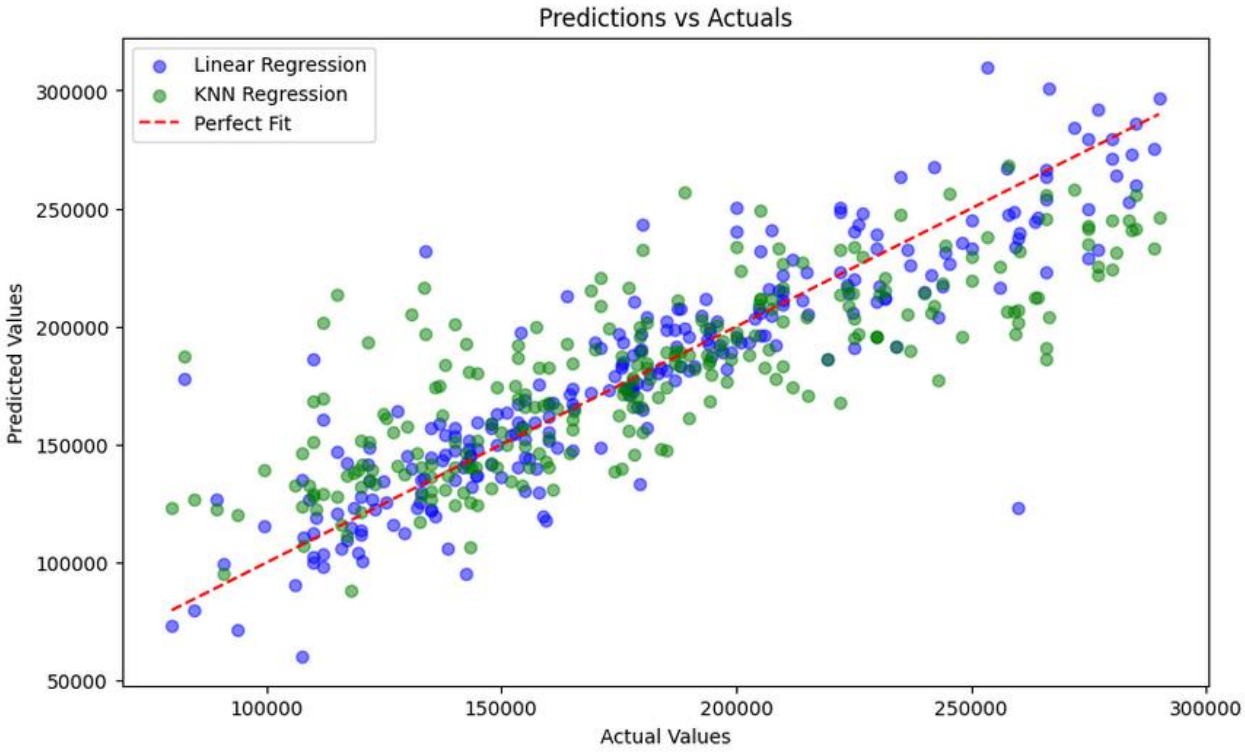
KNN training(k=5)

- We do the same splitting process

- **Mean Squared Error (MSE) for KNN Regression: 953809841.8384496**
- **Mean Absolute Error (MAE) for KNN Regression: 23696.493023255814**
- **R-squared: 0.6238049630461496**

Linear regression VS KNN

Model	MSE	MAE	R^2
Linear Regression	5.074068e+08	15375.789581	0.799872
KNN Regression	9.538098e+08	23696.493023	0.623805



The linear regression algorithm outperforms the KNN

Image Dataset

Dataset (The Oxford-IIIT Pet Dataset)

- It contains 38 classes of different species of dogs and cats but only 5 classes are used (project requirements).
- Each class contains 200 image (1000 total).
- Classes used in this model : (Bombay, Russian_Blue, basset_hound, great_pyrenees, pomeranian).

Images processing

- All images are resized to 64x64.
- The load_images function iterate over each class folder and the each image, resize them and returns two arrays, images contains the images loaded and labels contains class names.
- Images are reshaped to 1D vector
- Categorical labels are converted into numerical values using LabelEncoder()

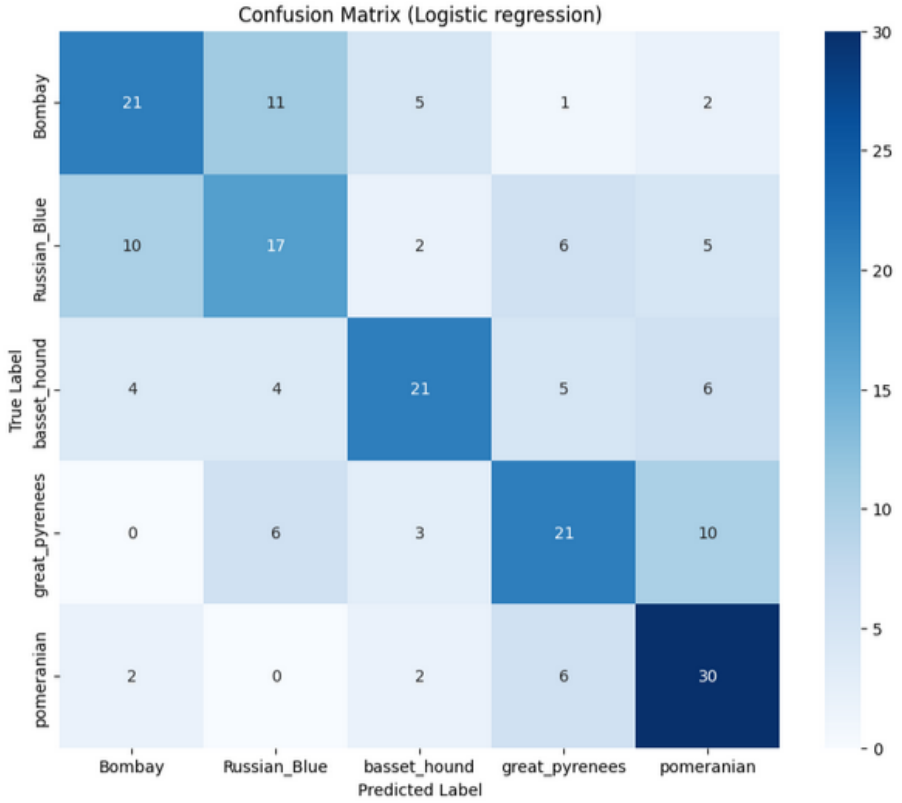
Logistic classification training

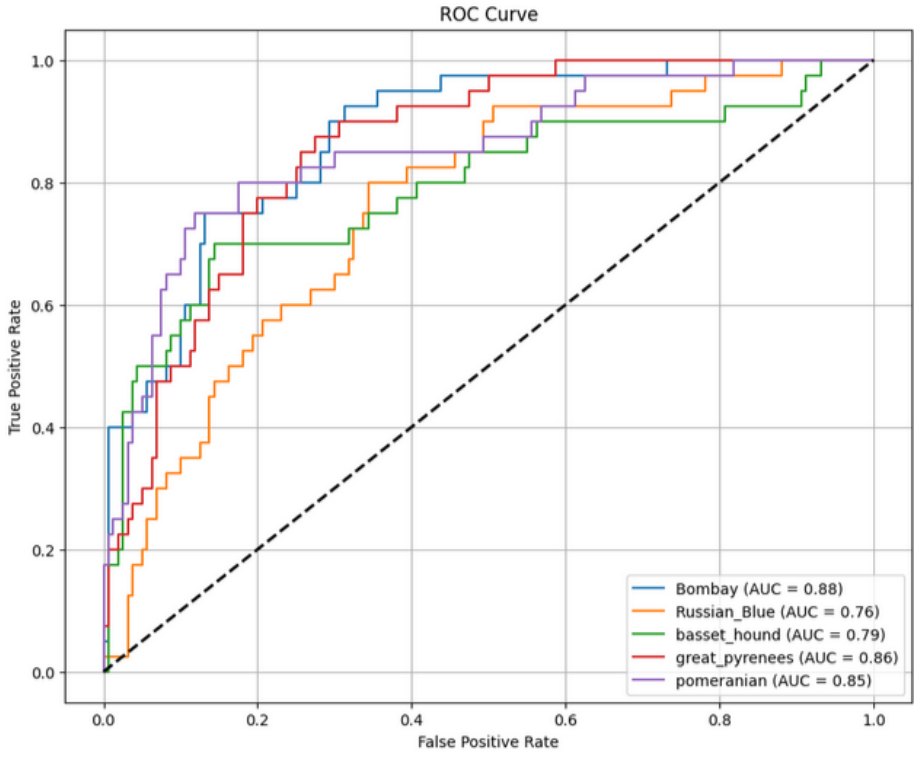
- We split the dataset into training and testing sets in terms of x and y.
- 20% are used for testing and the rest for training.
- The train and test sets are scaled to help the model treat all features fairly.
- We call the LogisticRegression function from sklearn and pass the training values to it.

- Accuracy: 0.55
- Loss: 2.2193

Classification Report for Logistic Regression:

	precision	recall	f1-score	support
Bombay	0.57	0.53	0.55	40
Russian_Blue	0.45	0.42	0.44	40
basset_hound	0.64	0.53	0.58	40
great_pyrenees	0.54	0.53	0.53	40
pomeranian	0.57	0.75	0.65	40
accuracy			0.55	200
macro avg	0.55	0.55	0.55	200
weighted avg	0.55	0.55	0.55	200


Confusion Matrix (Logistic regression)
True Label
Predicted Label
Bombay
Russian_Blue
basset_hound
great_pyrenees
pomeranian
21 11 5 1 2
10 17 2 6 5
4 4 21 5 6
0 6 3 21 10
2 0 2 6 30
30
25
20
15
10
5
0


ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Bombay (AUC = 0.88)
Russian_Blue (AUC = 0.76)
basset_hound (AUC = 0.79)
great_pyrenees (AUC = 0.86)
pomeranian (AUC = 0.85)

KNN classification training

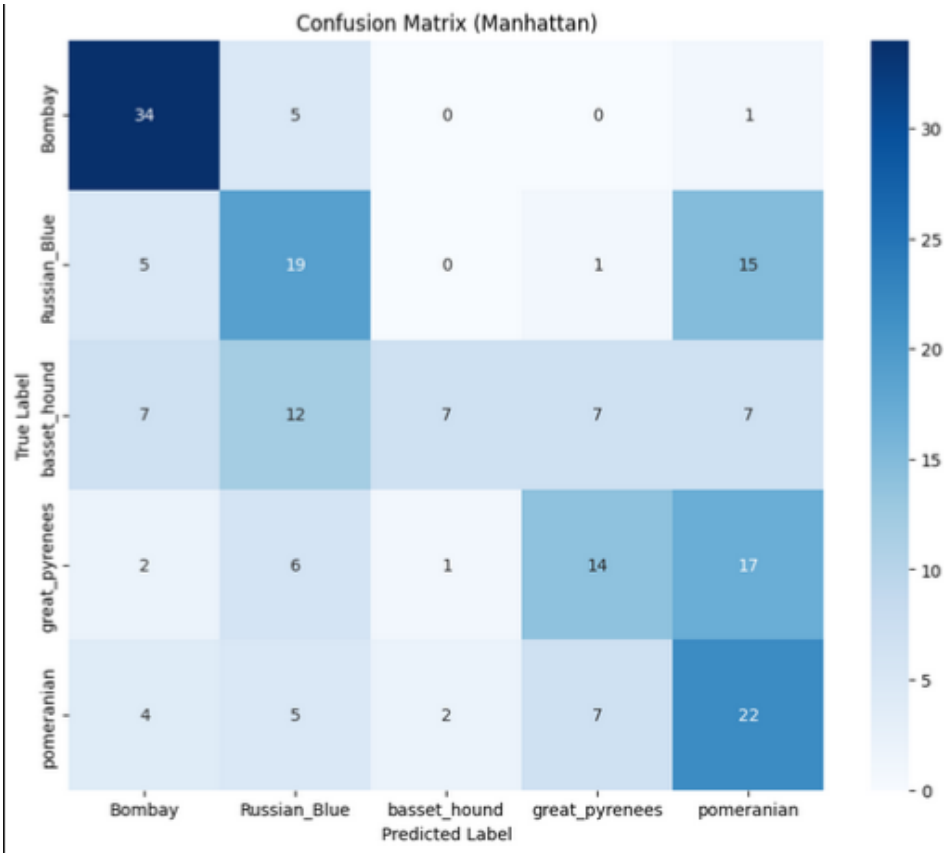
- Manhattan distance metric is used

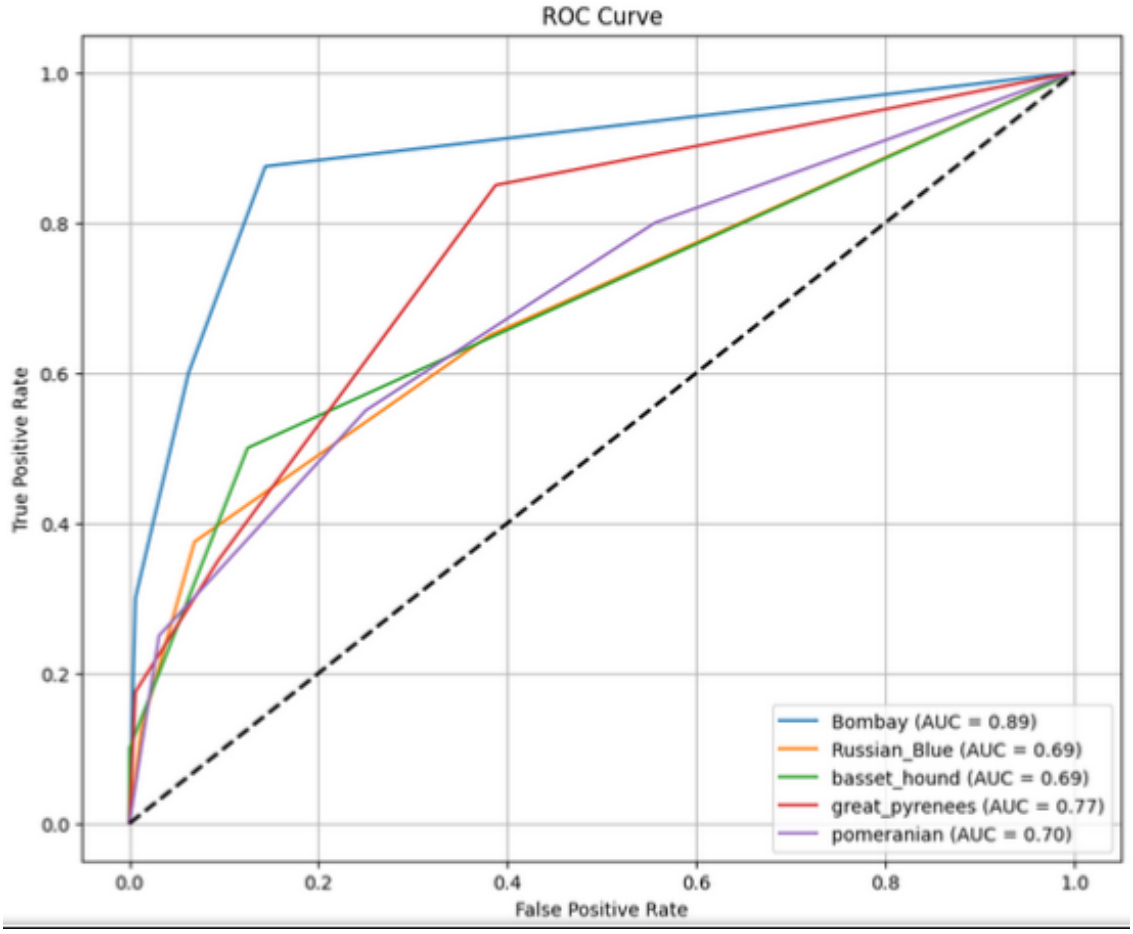
- **Accuracy**: 0.48

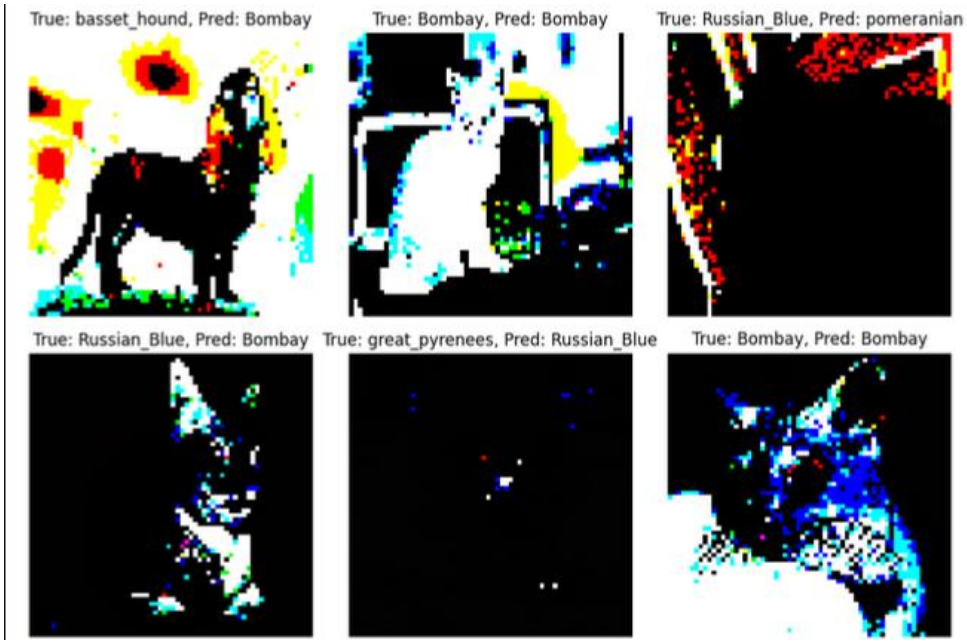
- **Loss : 0.52**

-

```
Classification Report for Manhattan Distance:
-----------------------------------------------
                precision    recall  f1-score   support

       Bombay       0.65      0.85      0.74        40
  Russian_Blue      0.40      0.47      0.44        40
  basset_hound      0.70      0.17      0.28        40
great_pyrenees      0.48      0.35      0.41        40
    pomeranian      0.35      0.55      0.43        40

      accuracy                           0.48       200
     macro avg      0.52      0.48      0.46       200
  weighted avg      0.52      0.48      0.46       200
```



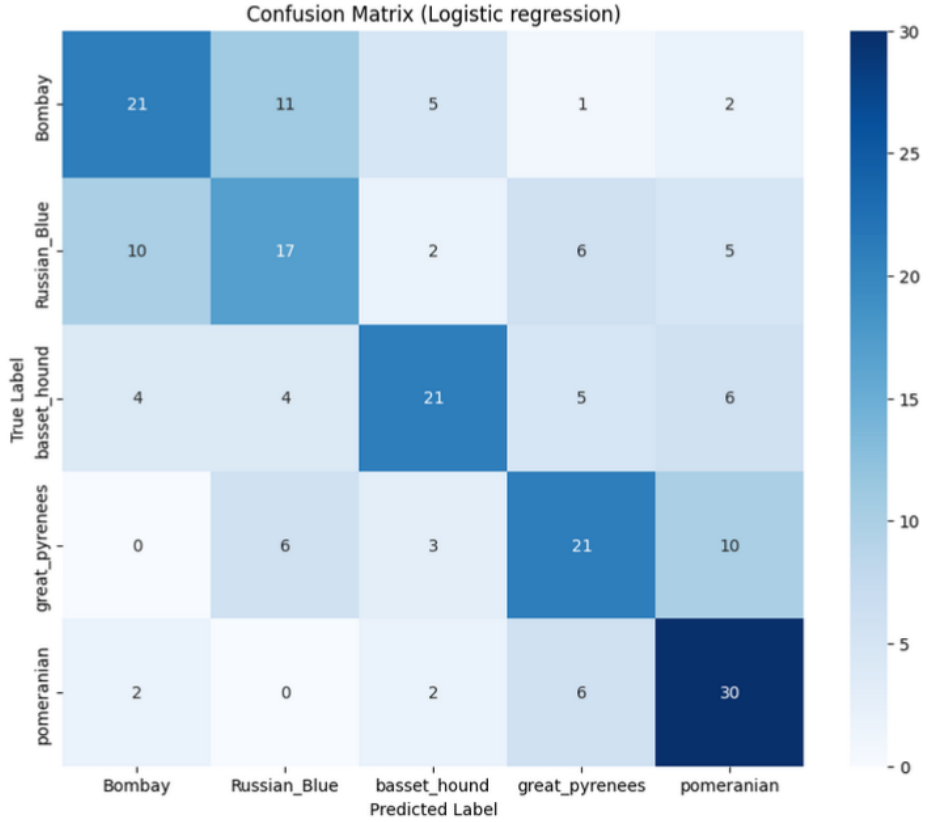

ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Bombay (AUC = 0.89)
Russian_Blue (AUC = 0.69)
basset_hound (AUC = 0.69)
great_pyrenees (AUC = 0.77)
pomeranian (AUC = 0.70)

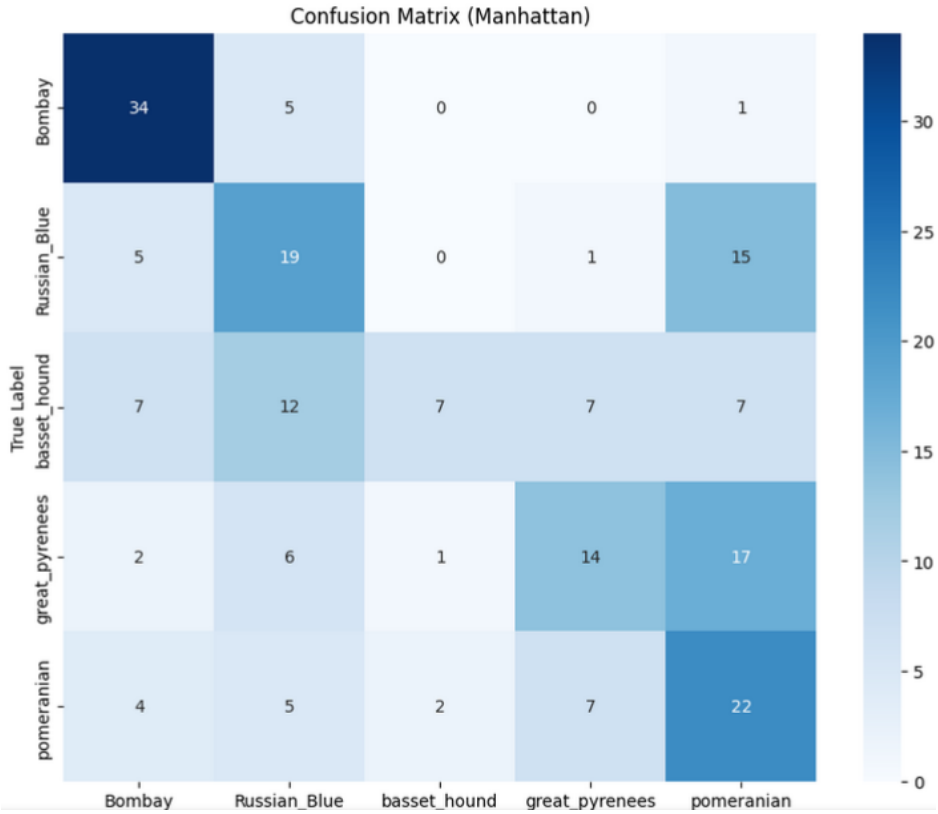

True: basset_hound, Pred: Bombay
True: Bombay, Pred: Bombay
True: Russian_Blue, Pred: pomeranian
True: Russian_Blue, Pred: Bombay
True: great_pyrenees, Pred: Russian_Blue
True: Bombay, Pred: Bombay

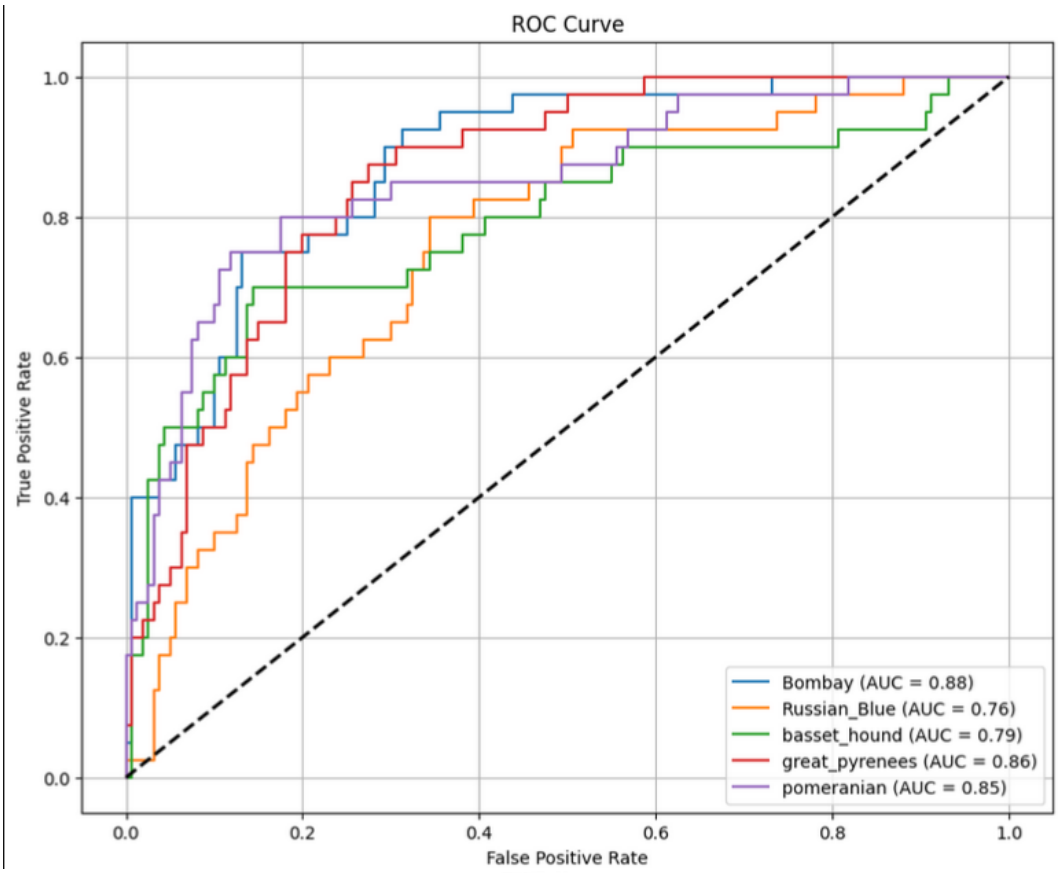
Logistic VS KNN

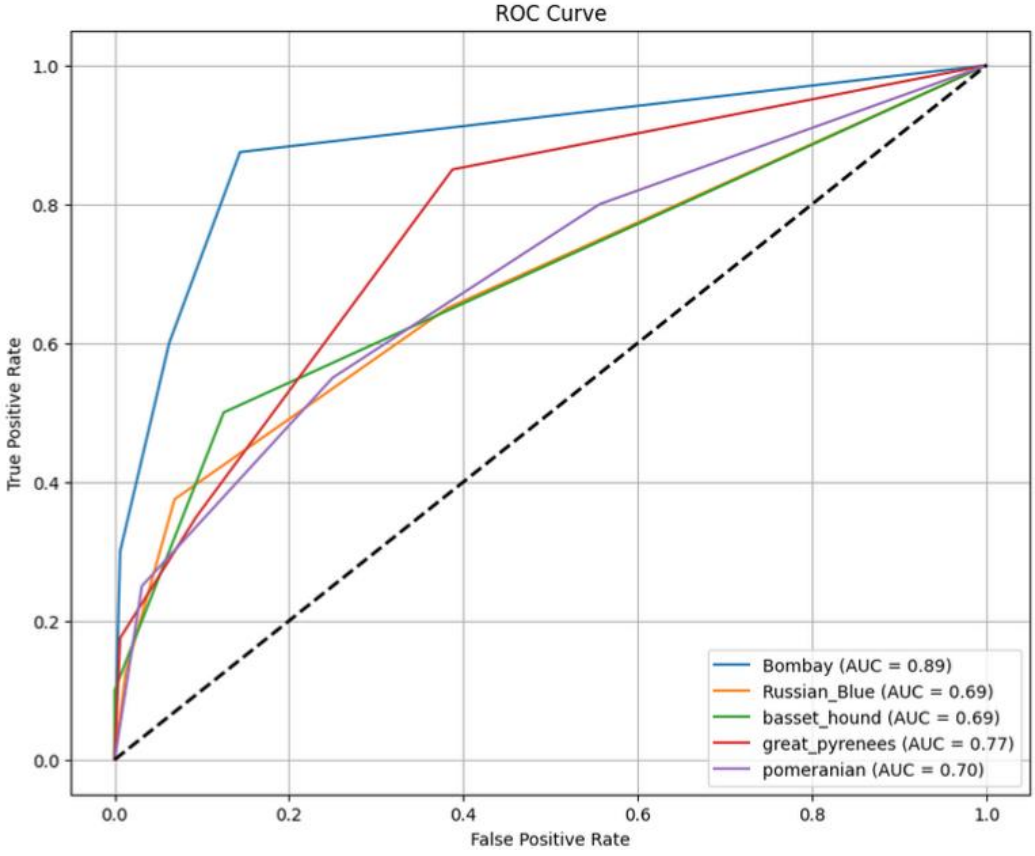
```
------------------------------------------------
Evaluating Logistic Regression
------------------------------------------------
Accuracy: 0.55
Loss: 2.2193

Classification Report for Logistic Regression:
------------------------------------------------
                precision    recall  f1-score   support

        Bombay       0.57      0.53      0.55        40
  Russian_Blue       0.45      0.42      0.44        40
  basset_hound       0.64      0.53      0.58        40
great_pyrenees       0.54      0.53      0.53        40
     pomeranian       0.57      0.75      0.65        40

      accuracy                           0.55       200
     macro avg       0.55      0.55      0.55       200
  weighted avg       0.55      0.55      0.55       200
```

```
Accuracy: 0.48
Loss : 0.52

Classification Report for Manhattan Distance:
------------------------------------------------
                precision    recall  f1-score   support

        Bombay       0.65      0.85      0.74        40
  Russian_Blue       0.40      0.47      0.44        40
  basset_hound       0.70      0.17      0.28        40
great_pyrenees       0.48      0.35      0.41        40
     pomeranian       0.35      0.55      0.43        40

      accuracy                           0.48       200
     macro avg       0.52      0.48      0.46       200
  weighted avg       0.52      0.48      0.46       200
```


Confusion Matrix (Logistic regression)
Bombay
Russian_Blue
basset_hound
great_pyrenees
pomeranian
True Label
Predicted Label
21 11 5 1 2
10 17 2 6 5
4 4 21 5 6
0 6 3 21 10
2 0 2 6 30
30
25
20
15
10
5
0


Confusion Matrix (Manhattan)
Bombay
Russian_Blue
basset_hound
great_pyrenees
pomeranian
True Label
34 5 0 0 1
5 19 0 1 15
7 12 7 7 7
2 6 1 14 17
4 5 2 7 22
30
25
20
15
10
5
0


ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Bombay (AUC = 0.88)
Russian_Blue (AUC = 0.76)
basset_hound (AUC = 0.79)
great_pyrenees (AUC = 0.86)
pomeranian (AUC = 0.85)


ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Bombay (AUC = 0.89)
Russian_Blue (AUC = 0.69)
basset_hound (AUC = 0.69)
great_pyrenees (AUC = 0.77)
pomeranian (AUC = 0.70)

The logistic regression algorithm outperforms the KNN in many classes