Isaac Kobby Anni

CS7300_F24_Assignment02

September 2024

1 Question One

Question one details the application of the **K-Nearest Neighbors (k-NN) algorithm** to the **Iris dataset**. The Iris dataset consists of three classes of flowers: **Setosa**, **Versicolor**, and **Virginica**, with 150 samples evenly distributed across the three classes, that is, 50 samples of each category. The objective is to classify these flowers into one of the three species based on their features. I used **F1 Score (Macro)** as the model evaluation metric due the nature of multi-class classification, ensuring that k-NN classifier performance unbiased toward other categories.
The dataset was split into training **(70%)**, validation **(20%)**, and test **(10%)** sets to properly evaluate model performance.

In the subsequent steps to select the best value of k for the k-NN model, I applied the elbow method, which involves plotting the F1 Score (Macro) for different k values (ranging from 1 to 15) for multiple distance metrics. The idea is to find the k value where the performance stabilizes or peaks. Four distance metrics were explored; **minkwoski**, **euclidean**, **manhattan** and **cosine**.

The plot below shows the F1 Score (Macro) for each combination of k and distance metric: From the elbow plot, the following observations can be made:

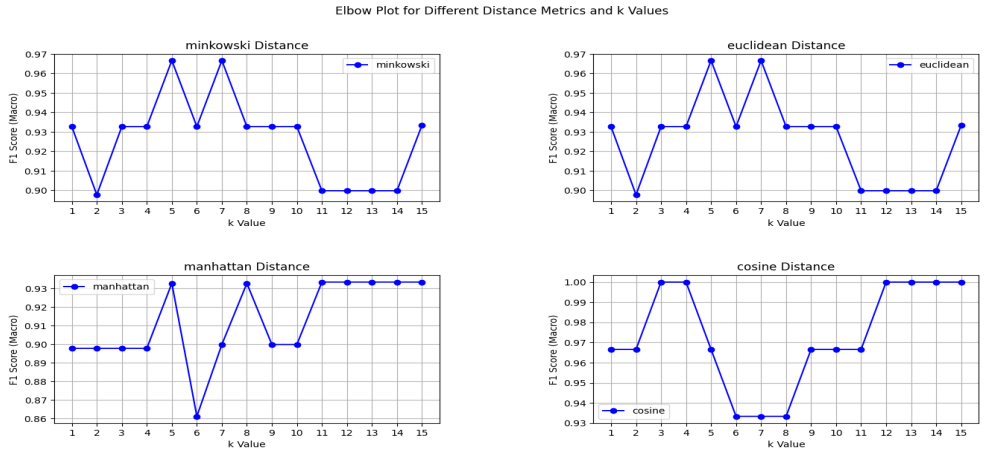

Figure 1: Elbow plot for best k

Minkowski Distance:

The F1 score peaks around k=5 and k=7 with values close to 0.97. After $k = 7$, performance declines sharply, especially beyond $k = 10$, indicating over-generalization as more neighbors are considered. The optimal k value for Minkowski distance is $k = 5$ or $k = 7$.

Euclidean Distance:

Similar to Minkowski, Euclidean distance performs best at $k = 5$ and $k = 7$, with F1 scores approaching 0.97. For k values greater than 10, the performance drops. Euclidean distance mirrors Minkowski's behavior, and the optimal k lies between 5 and 7.

Manhattan Distance:

The F1 score peaks at $k = 5$, but with slightly lower values compared to Minkowski and Euclidean (around 0.93). Performance stabilizes around $k = 7$ to $k = 10$, making Manhattan distance more stable for higher k values. Manhattan distance is more robust for larger k values, but its overall performance is slightly lower than Minkowski and Euclidean.

Cosine Distance:

Cosine distance performs exceptionally well, especially at smaller values of k such as $k = 3$ and $k = 5$, where it achieves an F1 score close to 1.0. The performance remains stable even for higher k values.

The bar plot below summarizes the performance across all distance metrics for k values between 1 to 15.

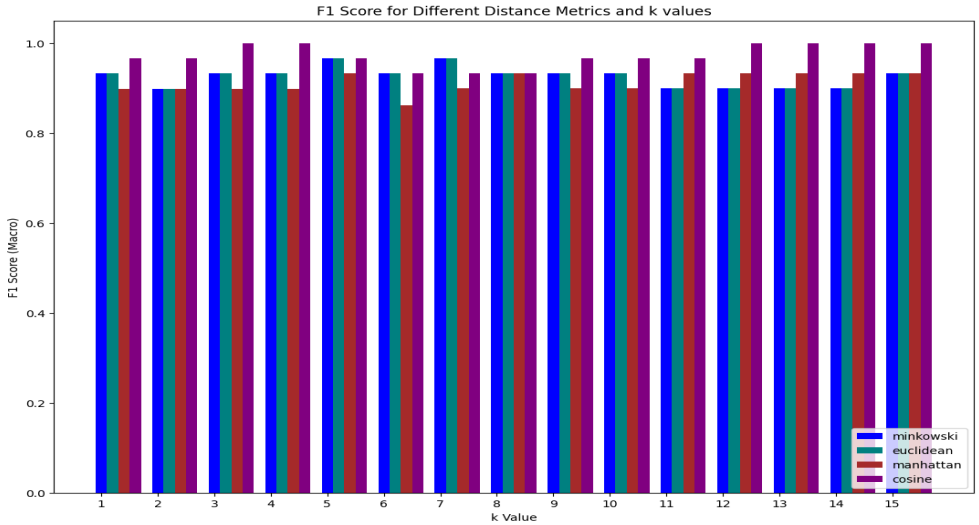

Figure 2: Bar plot for multiple distance metrics

Overall, based on the elbow plots and bar plot analysis, the best-performing k value lies between 3 and 5, with $k = 5$ being particularly strong across all metrics. Cosine distance is the top-performing metric, achieving near-perfect F1 scores at $k = 3$ and $k = 5$. Minkowski and Euclidean distances also perform well, with peak performance at $k = 5$ and $k = 7$, but their performance declines for larger values of k. Manhattan distance shows more stability for larger k values but has slightly lower performance compared to Cosine and Euclidean.

For this task, Cosine distance with $k = 3$ or $k = 5$ is recommended due to its superior performance across all metrics.

2 Question Two

In this part of the report, I applied **Ridge Regression** to a **housing dataset** and perform K-fold cross-validation and to determine which number of fold gives the best performance, that is, the minimum mean squared error. I also analysed the importance of the regression variables in the ridge regression model.

The dataset contains **333 samples and 14 features**, including predictors such as crime rate **(crim)**, number of rooms **(rm)**, and the target variable median value of owner-occupied homes **(medv)**.

First the training data was splitted into 70% and 30% for cross validation, fitted the model on the entire training set and then tested of the fitted model on the test set. Then finally made predictions on the *A1Q2_Test_Data* sample provided.

The table below summarizes the results for the cross validation approach.

k Value	Cross-Validation Score (MSE)
5	28.71
10	29.35
15	29.67
20	28.98
25	29.52
30	29.12

Table 1: Cross-Validation Mean Squared Error (MSE) for different values of k

As can be seen from the table the optimal k from the cross-validation appraoch is $k = 5$, which has the lowest meam-squared-error.

After training the Ridge regression model, I analyzed the feature importance by inspecting the coefficients learned by the model. Features with larger absolute coefficients have a greater influence on the target variable.

The plot below shows the importance of various features. The feature **lstat** (percentage of lower-

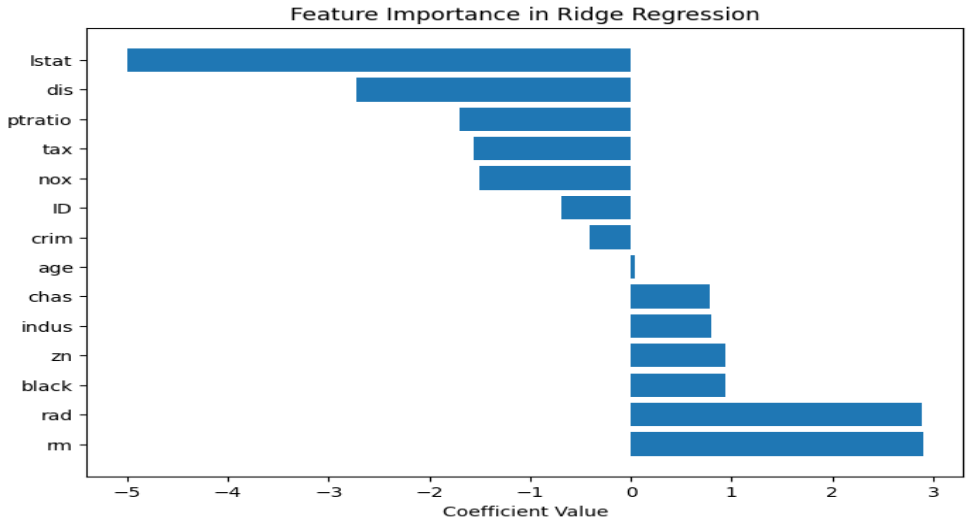

Figure 3: Feature importance plot

status population) has the largest negative influence on the target variable (medv), while **rm** (average number of rooms) has the largest positive influence, indicating that homes with more rooms are generally more expensive.

The choice of $k = 5$ for cross-validation is because it provided the minimum MSE compared against the other k-fold values.

3 Question Three

What is the curse of dimensionality and list three ways to avoid it?

This refers to various problems that arise when the number of features (dimensions) in a dataset increases. These issues include increased sparsity of data, which can make models prone to overfitting and less generalizable, as well as higher computational complexity. The following are three ways to avoid it.

- **Feature Selection:** Choose only the most relevant features using methods like Lasso regression, mutual information, or recursive feature elimination.
- **Regularization:** Apply regularization techniques such as Ridge or Lasso regression to penalize large coefficients and reduce model complexity.
- **Dimensionality Reduction:** Use techniques like Principal Component Analysis (PCA) or Singular Value Decomposition (SVD) to transform the dimension from a larger to a lower size however still retaining much information about the dataset.

Explain the differences between Ridge penalty and Lasso penalty.

Ridge Penalty (L2 Regularization): Adds a penalty equal to the sum of the squared values of the coefficients $\alpha \sum \omega_i^2$. It shrinks the coefficients but does not set any of them to exactly zero. Ridge is useful when dealing with multicollinearity and keeps all features in the model.

Lasso Penalty (L1 Regularization): Adds a penalty equal to the sum of the absolute values of the coefficients $\alpha \sum |\omega_i|$. It can shrink coefficients to zero, effectively performing feature selection by excluding less important features from the model.

Demonstrate your insights for both direct and iterative methods for optimization.

Direct Methods: These methods attempt to solve optimization problems analytically in one step. For example, Ordinary Least Squares (OLS) in linear regression solves for the coefficients by directly calculating the inverse of the matrix. They are efficient for small, well-structured problems but can become computationally expensive for large datasets due to matrix inversion.

Iterative Methods: These methods, like Gradient Descent, start with an initial guess (as random initialization) and iteratively improve the solution by taking steps proportional to the gradient of the objective function. They are preferred for large datasets or when direct methods are computationally expensive. Iterative methods are more flexible but require careful tuning of hyperparameters like learning rate and can take more time to converge in most cases or not converge at all.