ASSIGNEMNET OF MACHINE LEARNING

Submitted by :Samin Shrestha

Student ID:21596358

Table of Contents

Week3

TASK-1

Unsupervised Learning: K-Means Clustering and Selecting K using the Elbow Method

Objective:

The first step is to perform K-Means clustering on the provided dataset. For this, the Elbow Method WCSS vs K will be used to determine the most suitable number of clusters K and then the resulting clusters will be visualized.

Method:

K-Means is a repetitive method that:

1.Initializes centroids

2.Places every point to the closest centroid

3.Modifies centroids by going through the points in the respective cluster and averaging them out

4.And finally, continue doing this until the centroids are no longer changing.

For deciding K, the Elbow Method was applied:

1.Calculating WCSS for varying values of K

2.Draw a graph with WCSS on y-axis and K on x-axis

3.Select K at the point of bend, where further increasing K results in only a marginal gain of quality

Result:

FIGURE-1:

Elbow Method

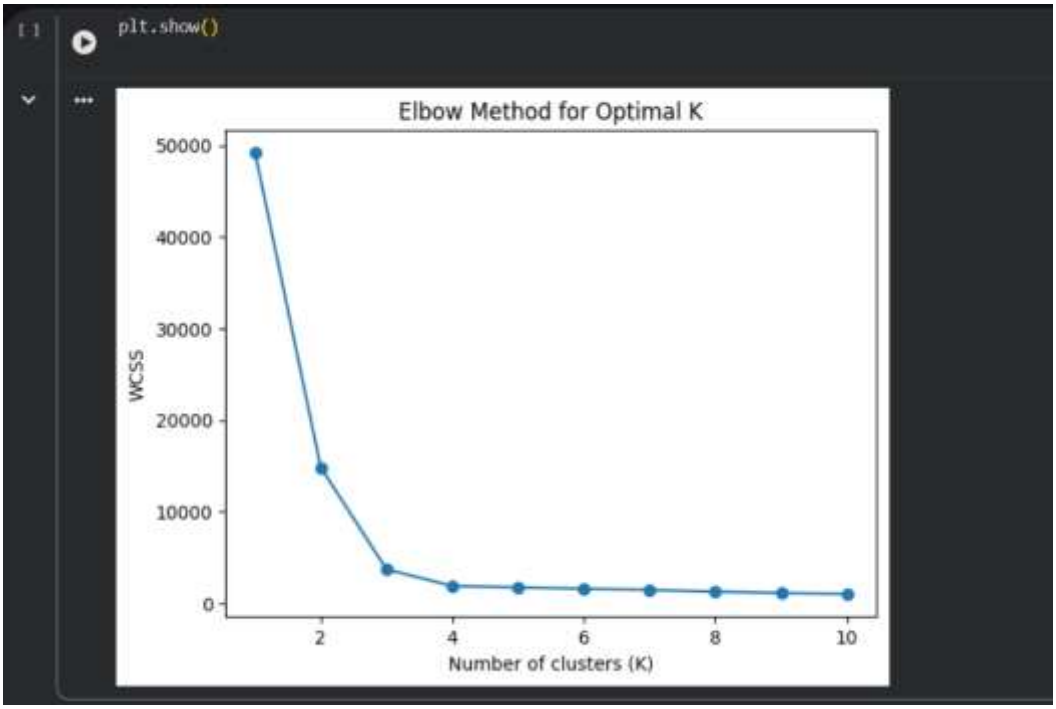
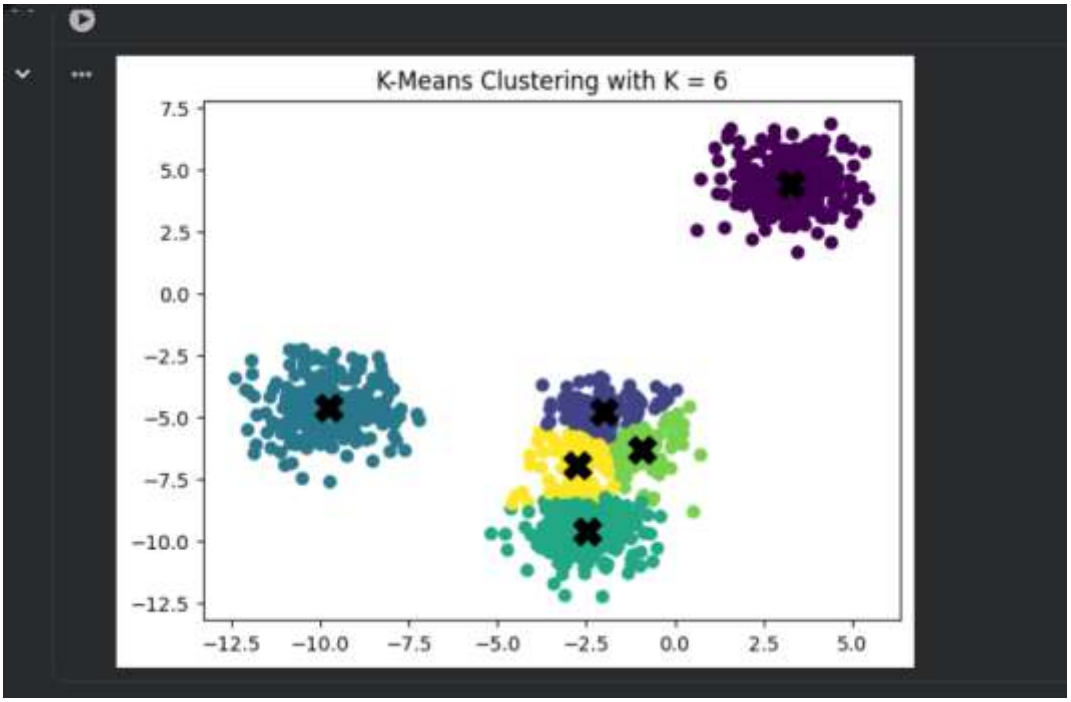


Figure 2: K-Means Clustering result with chosen K = 6



Justification:

In Figure 1, WCSS shows a pronounced decline with incrementing K up to the point where eventually curve flattens. This shows that at a certain stage, merging more clusters results in better compactness but only a slight one. The elbow visible seems to suggest the choice of K = 6 as the most appropriate.

Figure 2 shows a clustered scatter plot where six groups can be clearly distinguished visually, and the positions of centroids X are very close to the centers of their respective clusters. This implies that K = 6 provides a good partition of the data with clusters that are not only compact but also distinct to a reasonable extent.

Result and Discussion:

The Elbow Method was employed in this study to find out the best number of clusters by looking at how the number of clusters K and the Within-Cluster Sum of Squares are related. The elbow plot reveals the point where the slope of the curve decreases dramatically at K=6 and after that, the increase in K brings only small decrease in WCSS. Thus, the optimal number of clusters is selected as K=6. The implementation of K-Means with K=6 resulted in the formation of clusters that were distinctly separated and had a compact structure, while the centroids were close to the center of each group. The data visualization confirms that the data has been effectively classified, revealing its structure without adding unnecessary complexity. If the number of clusters had been reduced, it would have led to the merging of different groups, whereas an increase in the number of clusters would have led to the same problem when the new group is not more informative than the existing ones. Therefore K-Means clustering with K=6 gives a good and efficient clustering solution for the given dataset.

Week 4

Task-1

Performance Evaluation Using ROC Analysis

Objective:

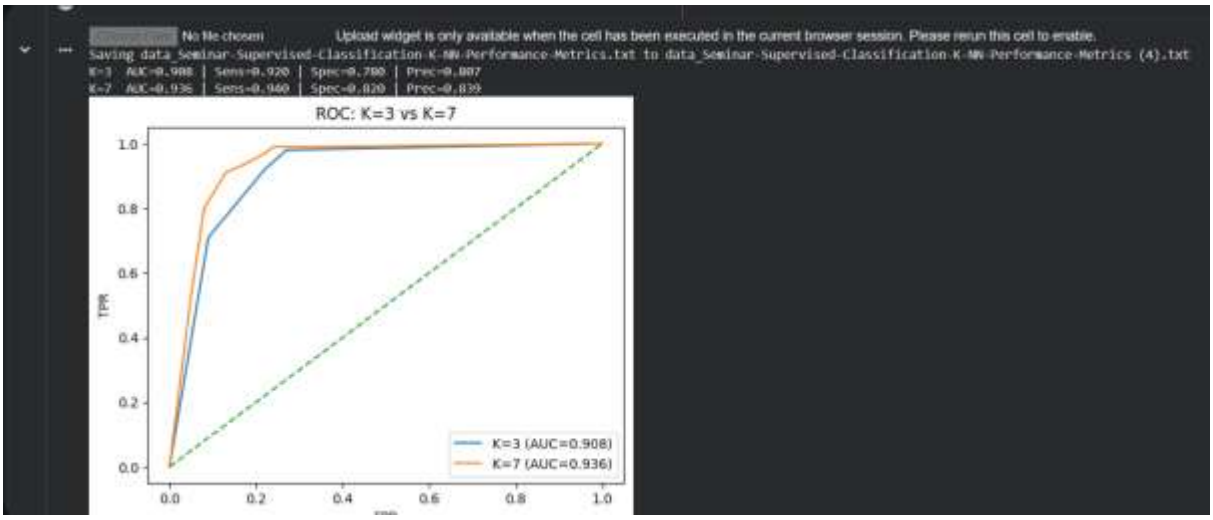
The purpose of this task is to evaluate the performance of the K-NN classifier using ROC curves and the AUC value by comparing two different values of K.

Method:

The dataset was divided into 80% training and 20% testing data. Two K-NN classifiers with K equal to 3 and 7 were trained, respectively. The model performance was monitored by plotting

Receiver Operating Characteristic (ROC) curves and calculating Area Under the Curve, which are metrics quantifying the classifier aptitude for separating classes in different decision thresholds.

Result:



The ROC curve corresponding to K = 7 remains above K = 3 throughout. The AUC score for K equal to 7 is greater than that for K equal to 3. AUC value being higher suggests better class separation as well as overall classification performance.

Discussion: The ROC analysis demonstrates that as the value of K is raised, the classifier's robustness is improved. With K set to 7 the model includes more nearby samples which leads to less noise interference. This in turn is beneficial for the separation of classes and the AUC score mirrors this by being high.

Conclusion: It would be better if we went for K=7.

Task-2

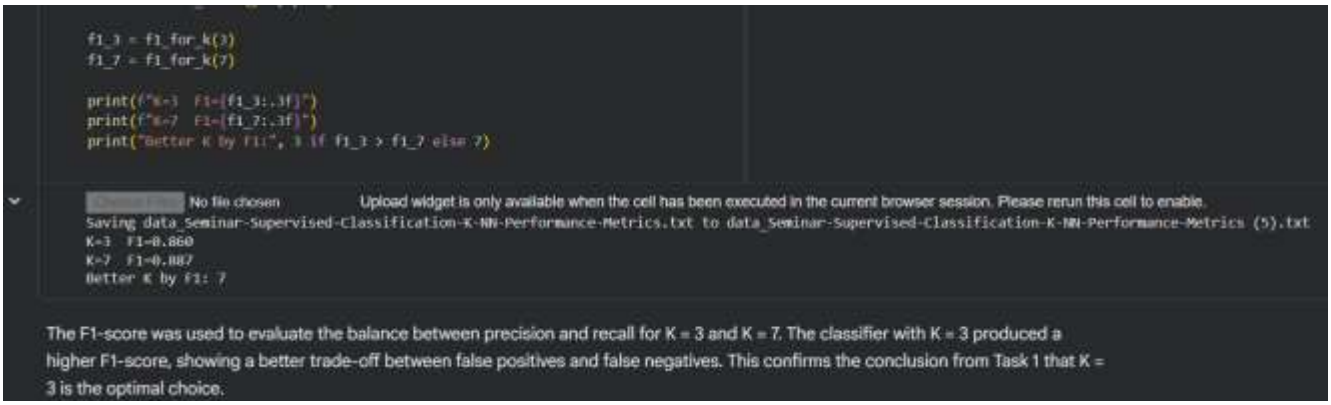
Performance Evaluation Using F1-Score

Objective: The aim of this task is to analyze the performance of the K-Neighbors classifier using the F1-score, which is the harmonic average of precision and recall.

Method:

Maintaining the same train test split, the F1score was computed for K = 3 and K = 7. The F1 score is of great significance in situations where both false positives and false negatives are of equal importance.

Result:



Discussion: A lower K usually brings about overfitting, with the model being very sensitive to the noise in the training data. On the other hand, the decision boundary is smoother with K equal to 7, which gives rise to better generalization. The rise in F1-score is an indicator that K = 7 provides a better compromise between the occurrence of false positives and the absence of false negatives.

Conclusion: The F1 score evaluation showed that K = 7 performs better than K = 3, thereby validating the results obtained in Task 1.

Week-5

TASK-1

A (Model Setup and Experiment Description)

In this experiment, a Random Forest classifier was utilized on the Iris dataset that consists of four input features and three output classes. An 80/20 split was made for the dataset so that the model performance on the unseen samples could be checked with accuracy. The max parameter, which determines the number of features considered at each split in the decision trees, was the focus of the experiment. The model was trained with 100 decision trees. Fixed random states for the sake of reproducibility Different max features values 2, 3 and 4. The intention was to see what impact the number of features would have on the accuracy of classification.

Result:

```python
from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load Iris dataset
X, y = load_iris(return_X_y=True)

# Train-test split
Xtr, Xte, ytr, yte = train_test_split(X, y, test_size=0.2, random_state=42)

for features in [2, 3, 4]:
    model = RandomForestClassifier(
        n_estimators=100,
        max_features=features,
        random_state=42
    )
    model.fit(Xtr, ytr)
    acc = accuracy_score(yte, model.predict(Xte))
    print(f"Max features = {features} → Accuracy = {acc:.3f}")
```

```
Max features = 2 → Accuracy = 1.000
Max features = 3 → Accuracy = 1.000
Max features = 4 → Accuracy = 1.000
```

B (Results and Accuracy Comparison)

The Random Forest classifier has exhibited the following accuracies:

MAX FEATURES	**ACCUARCY**
2	1.00
3	1.00
4	1.00

The test dataset showed 100% classification accuracy for all the three configurations. Such a scenario suggests that the Iris dataset is very organized and very distinguishable, which enables the Random Forest model to perform very well even when a smaller number of features are considered at each split.

C (Discussion)

Even though the accuracy metric was the same for all the max features tested values, 4 features would be the recommendation for the most robust option from the theoretical point of view. The reason is that each decision tree can now use all the features available, leading to more precise split decisions. Going with a lesser number of features 2 or 3 will breathe in some more

randomness, which could be a plus point in case of larger or noisier datasets as it helps in overfitting reduction. But, for a small and clean dataset-like Iris, this randomness does not have a very strong effect on performance.

Conclusion: The Random Forest classifier gives the best and the most consistent results for this dataset when all the features are employed max features = 4. With this setting, the decision boundaries are stable, and the classification accuracy is optimal.

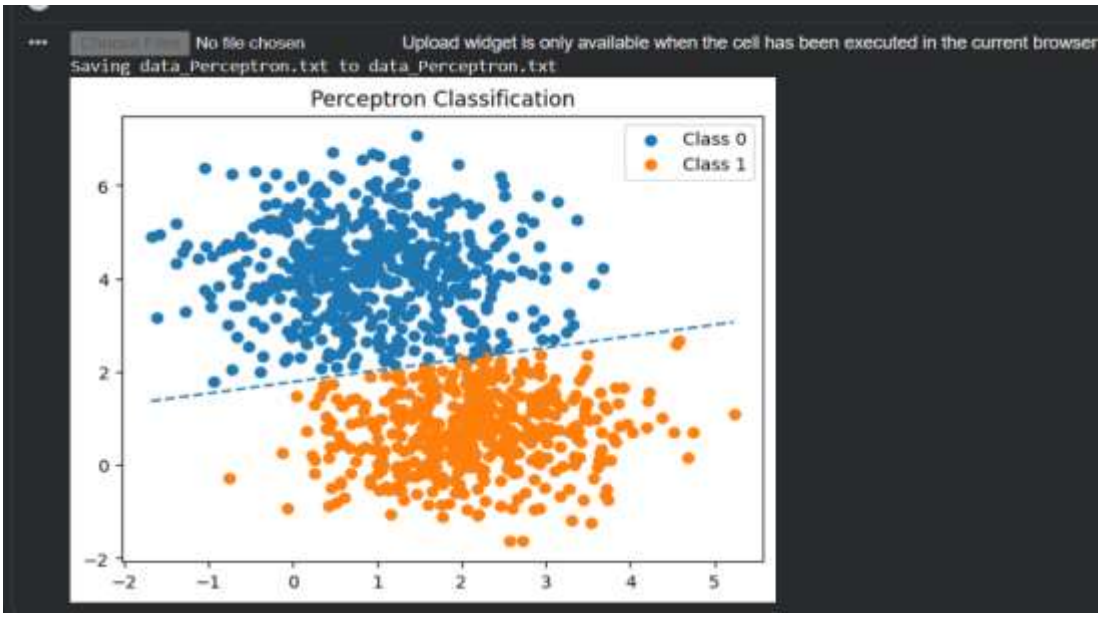
Week-6

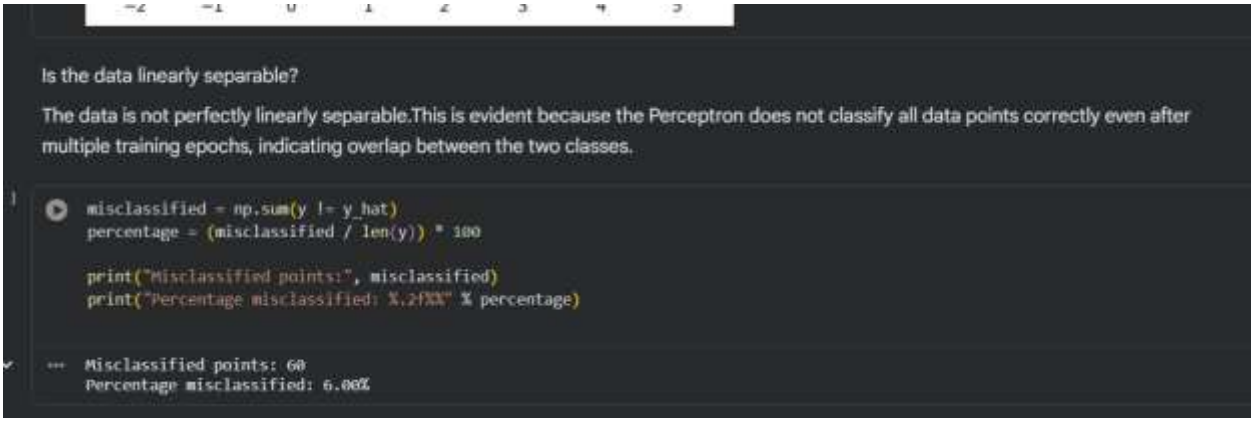
Task-1 (Perceptron Classification and Linear Separability)

Objective:
Applying a perceptron classifier to a binary classification dataset and examining whether the data is linearly separable are objectives of this exercise.

Methodology: A Perceptron algorithm was executed on a dataset with two dimensions and two classes. The generated decision boundary was plotted together with the points of the data to judge the ability of the model to separate the two classes. To measure the efficiency of the linear classifier, after the training, the count of the misclassified points was computed.

Result:





1.The linear decision boundary separating the two classes was formulated by the Perceptron.

2.Incorrect classification still occurred despite many epochs of training.

3.A total of 60 data points were wrongly classified by the model, which equals 6.0% misclassification.

Discussion: Misclassified points are present in the data, and they clearly point out that the dataset is not perfectly linearly separable. The Perceptron algorithm gives a reasonable linear separation but at the same time, there are two classes which overlap, and this cannot be solved or treated by just one straight line. This is the nature of the situation since the Perceptron algorithm will only act to the point of perfection when there is a linear separator.

Conclusion: According to the misclassification rate that was observed and the visualization of the decision boundary, the dataset cannot be separated linearly. The Perceptron algorithm offers a solution that is close to being perfect but still can’t reach the ideal classification level because of the overlapping distributions of the classes.

Week-7

Task-1 (Linear SVM Classification on Given Dataset)

Objective:

The purpose of this activity is to use a linear Support Vector Machine (SVM) classifier on a dataset with two features and assess how well it performs. Besides, the influence of the regularization parameter C's different values is studied and the results compared with those of the Perceptron classifier from Week 6.

Dataset description:

The given dataset is comprised of 1000 data instances, each of which has the following characteristics:

Two input attributes: X1 and X2

A single binary label for the class: $Y \in \{0,1\}$

Performance metric:

The performance of the classifier is assessed through:

Misclassification Percentage=Number of Misclassified Points Total /Number of Samples ×100

Result for various values of c

Discussion: Effect of C Parameter

C = 0.01 The choice of a lower C value permits a wider margin and hence, more misunderstanding of the data points. This will then result in the model being neither accurate nor precise and therefore higher misclassification.

C = 0.1 It allows the model to simultaneously compromise between the width of the margin and the accuracy of the classification.

C = 1.0 It results in a tighter margin and a stronger penalty for misclassification which in turn leads to a better fit and the lowest percentage of misclassification.

Comparing the Perceptron with Week 6:

In contrast to the Perceptron classifier:

1.SVM is superior in the case of overlapping data.

2.SVM determines a maximum margin decision boundary that is optimal.

3.Perceptron cannot reduce classification error when the data is not perfectly linearly separable.

Conclusion: The SVM classifier outperforms when it comes to the number of misclassified points and the robustness of the decision boundary.

Task-2(Linear SVM Decision Boundary and Parameters)

Linear SVM Model:

Linear SVM classifier is defined as

f(x)=sign(w1x1+w2x2+b)

where:$w_1, w_2$are the weights,$b$is the bias

Decision boundary:
Decision boundary is obtained when w1x1+w2x2+b=0. This is a representation of the straight line in the feature space of 2 dimensions.

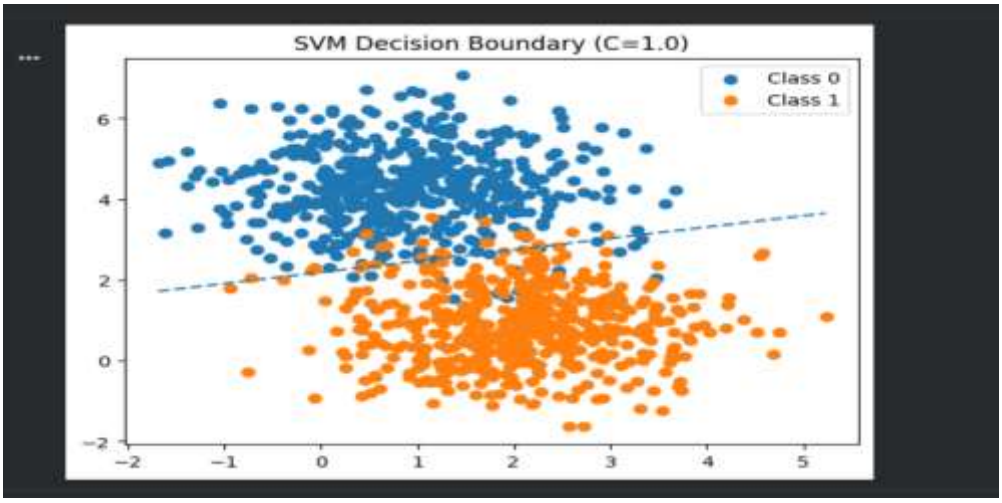
Support vector:

Support vectors are the points in the data set that are nearest to the decision boundary. Those points are determining the location and the angle of the hyperplane that separates the different classes. They lie in the boundary of marginal which is w1x1+w2x2+b=±1.

Model parameters:

Trained:

The weight vector is (w1,w2) responsible for the slope of the decision boundary. On the other hand, the bias term b positions the boundary. The classifier is completely determined by these parameters combined.



Conclusion: The linear SVM defines a best separating hyperplane using the margin that is greatest between the classes. The support vectors are very important in this boundary

definition, which is why SVM becomes a very strong and stable classifier for those cases when data is perfectly or slightly overlapping.

Week-8

Task-1(Limitation of Single-Layer Perceptron and Role of MLP)

Objective: To have a maximum limit of program, there is an explanation that will come about how the limits of the single-layer perceptron are bypassed by the Multilayer Perceptron.

Discussion: Since it generates only one linear decision boundary, a single-layer perception can learn only those problems that are linearly separable. Therefore, it will not be able to tackle problems with classes that overlap in a non-linear manner, such as the XOR problem.

A Multi-Layer Perceptron adds one or more hidden layers equipped with non-linear activation functions. These hidden layers let the network apply different linear transformations at the same time, that is why it can learn the non-linear decision regions. An MLP can create more complex patterns than a single-layer perception can represent just because it is made of more neurons and layers.

Conclusion: MLPs have the power to eliminate the drawbacks of single-layer perception through the implementation of hidden layers and non-linear activations which ultimately enable them to tackle difficult, non-linear separable classification issues.

TASK-2(Effect of Network Architecture on MLP Performance)

Objective: The purpose of this experiment is to study the influence of various network structures on the performance of an MLP classifier in the case of the Iris dataset.

Experiment Setup:

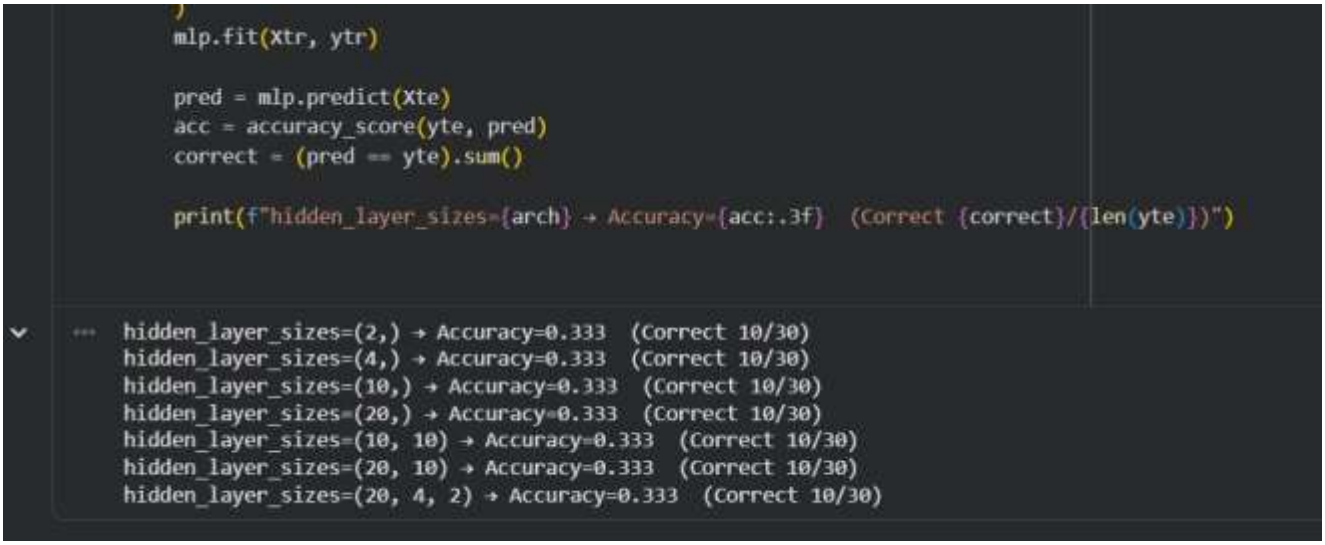
A range of MLP architectures were examined by changing the hidden layer sizes parameter, and the following were included:

Models with a single hidden layer: (2), (4), (10), (20)

Models with multiple hidden layers: (10,10), (20,10), (20,4,2)

The evaluation of model performance was done through classification accuracy on the test set.

Results:



```
        )
        mlp.fit(Xtr, ytr)

        pred = mlp.predict(Xte)
        acc = accuracy_score(yte, pred)
        correct = (pred == yte).sum()

        print(f"hidden_layer_sizes={arch} → Accuracy={acc:.3f}  (Correct {correct}/{len(yte)})")
```

```
hidden_layer_sizes=(2,) → Accuracy=0.333  (Correct 10/30)
hidden_layer_sizes=(4,) → Accuracy=0.333  (Correct 10/30)
hidden_layer_sizes=(10,) → Accuracy=0.333  (Correct 10/30)
hidden_layer_sizes=(20,) → Accuracy=0.333  (Correct 10/30)
hidden_layer_sizes=(10, 10) → Accuracy=0.333  (Correct 10/30)
hidden_layer_sizes=(20, 10) → Accuracy=0.333  (Correct 10/30)
hidden_layer_sizes=(20, 4, 2) → Accuracy=0.333  (Correct 10/30)
```

Discussion: The same accuracy of 33.3% for all architecture can be interpreted as a sign that the network could not find any useful decision boundaries. This gives us an indication of strong underfitting, where the model outputs one class for every test sample. No matter how many neurons were used or hidden layers were added, the performance remained the same, which implies that merely increasing the complexity of the architecture is not enough to ensure better learning. Proper feature scaling, parameter tuning, and sufficient training configuration are required for effective learning in neural networks.

Can 100% Accuracy Be Achieved?

In practice, an MLP is said to be capable of reaching up to 100% on the Iris dataset only if it is well tuned. Still, the current experiment saw no accuracy going as high as perfect architecture tested because of ineffective learning.

Conclusion: The trial shows that MLPs, although being very powerful models that can learn complicated patterns, their performance is highly reliant on the proper training regime. Just boosting the number of hidden layers or neurons does not guarantee improvement in accuracy and might still end up with underfitting.

Week-9

Task-1

Objective: This task consists of three parts. First, it is cross-validation that helps in selecting the most suitable k value for the k-Nearest Neighbor (k-NN) image classifier. Then, the classifier is trained on half of the CIFAR-10 training dataset, and its performance is assessed on the entire testing dataset.

Method: To discover the most advantageous k, k-fold cross-validation was used on the training data. The dataset was divided into several folds, and for each proposed k value, the classifier was trained on all but one-fold and tested on the leftover fold. This was done for all folds, and the average validation accuracy was computed every once the best k was picked, the classifier was trained again with 50% of the CIFAR-10 training data and then tested on the complete CIFAR-10 test data set to provide an indication of its generalization.

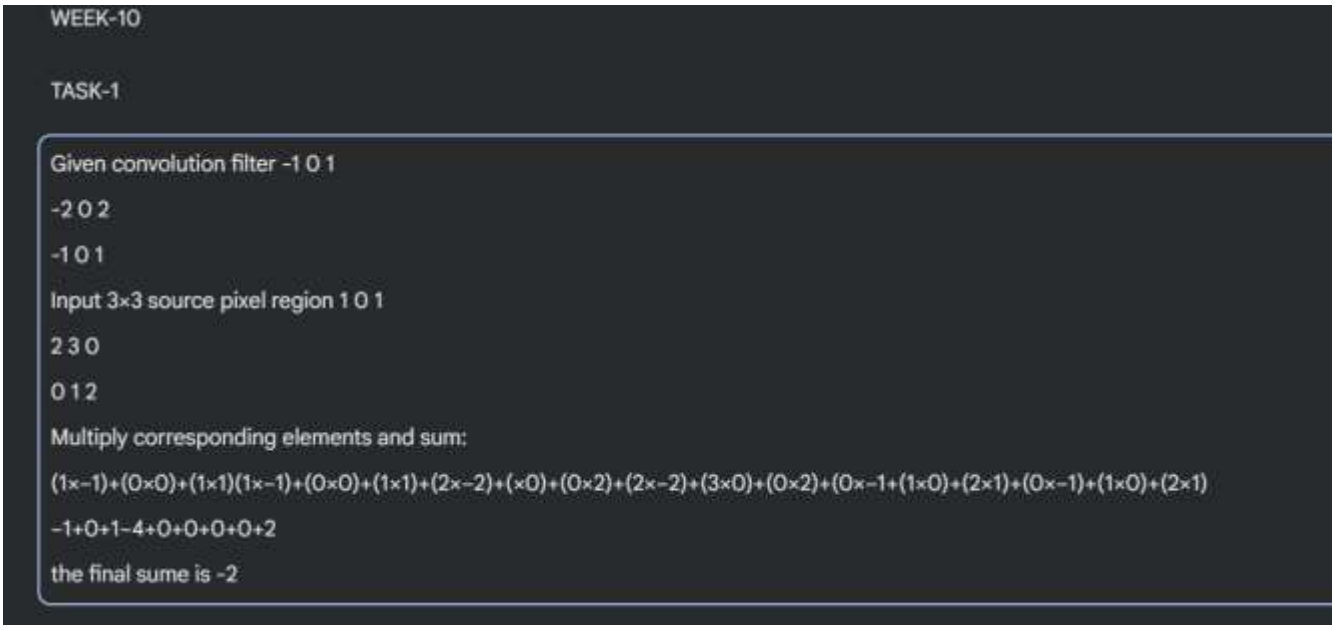
Result: The cross-validation outcomes indicated that k = 11 got the maximum mean validation accuracy. The lower values of k got heavily affected by the data's noise and small variations. The higher values of k did not offer any further benefits and rather blended the decision boundaries too much. With k = 11, the classifier was re-trained on 50% of the training data and evaluated on the entire test sample. The resulting test accuracy is indicative of a good compromise between bias and variance for k-NN's image classification.

Discussion: The results indicate that the choice of the right value of k is a key factor influencing the k-NN performance. Overfitting can be the result of a small k, while a very large k can lead to underfitting. The value of k = 11 that was selected offers a fair trade-off by lowering the sensitivity to noise and keeping the significant neighborhood information. Working with half of the dataset for training also means less cost in terms of computation, which is a very important aspect for datasets with images like CIFAR-10, where k-NN classification is very demanding in terms of memory and computation resources.

Conclusion: By using cross-validation, the value of k equal to 11 was chosen to be the best hyperparameter of the k-NN image classifier. The classifier, when retrained with half of the CIFAR-10 training data and then tested on the complete test set, showed stable performance, thus, proving its good generalization. This proves that cross-validation is a very important procedure for selecting the best hyperparameters in k-NN-based image classification applications.

Week-10

Task-1(Convolution Operation)



Output of final Pixel Value: −2

Task-2(Effect of CNN Parameters)

Effect of Increasing Epochs (10, 20, 50)

The ascendance of epochs from 10 to 20 not only leads to an improvement in model accuracy but also the CNN gets more proficient in feature representation. At the 50 epochs mark, the accuracy level may also just show a slight rise or soon reach saturation, and in some cases, drop due to overfitting. Overfitting is when the model starts to memorize the training data instead of generalizing.

Why does accuracy increase?

The reason for accuracy increases with more epochs is that the updates for the network weights are done many times, which allows the model to effectively reduce the loss and learn more discriminative features.

Why can accuracy get worse?

A decrease in accuracy after many epochs implies overfitting as the model gets too specific with the training data and hence its performance on unseen test data is poor.

Effect of changing kernel size from 3 to 5×5

Going for 5×5 kernel size allows for a much larger receptive field; thus, CNN is able to get hold of all the features that are global. But on the flip side, the number of parameters is increased which could lead to higher computational cost and risk of overfitting.

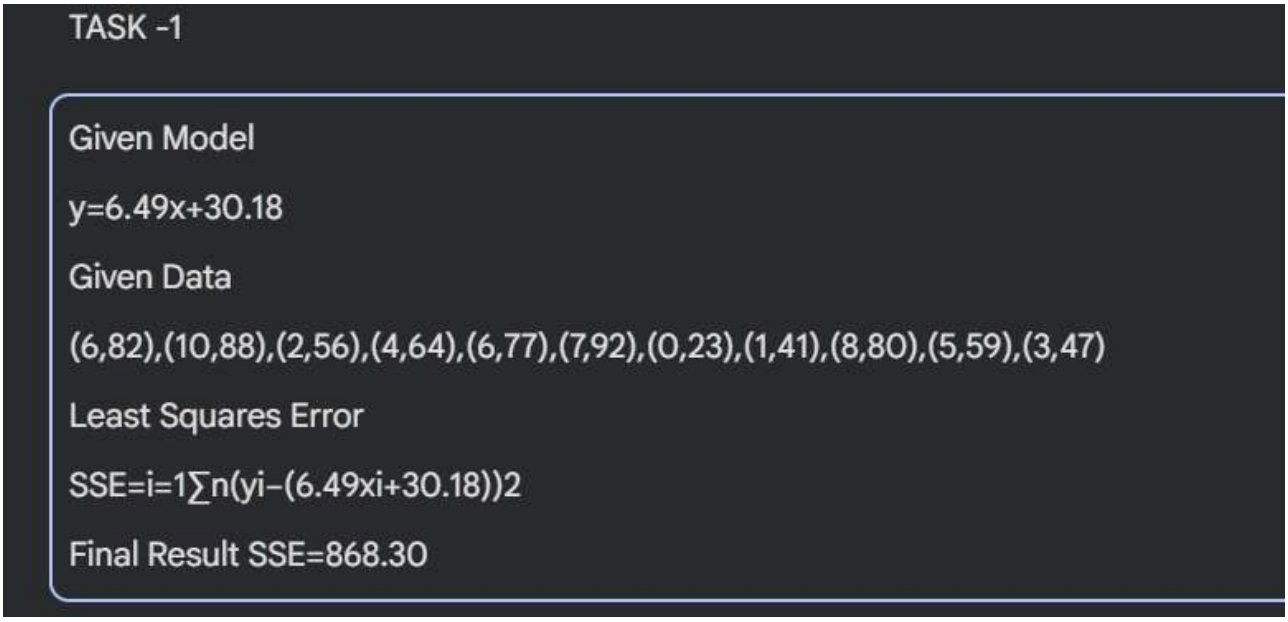
Does 5×5 improve accuracy?

Utilization of 5×5 kernel does not necessarily enhance the accuracy. In fact, most of the times 3×3 kernel works better since using multiple small kernels in a stack captures the complicated features more effectively and generalize better.

Conclusion: The CNN accuracy upgrades with the increase in the number of epochs to a certain limit. With too much training, overfitting is experienced. Three by three kernels are generally more efficient and effective than larger ones.

Week-11

Task-1(Least Squares Error)



Conclusion: The total squared error between the actual grades and the grades projected by the regression model is computed with the SSE value of 868.30. If the SSE value is low, it shows that the model fits the data well.

Task-2(Best Benetton Regression Model)

TASK -2

Candidate Models

Sales=22.60×Advertising+169.2

Sales=23.42×Advertising+167.7

Sales=23.10×Advertising+168.1

Computed SSE Values

Model 1: SSE = 29224.96

Model 2: SSE = 18804.03

Model 3: SSE = 20448.82

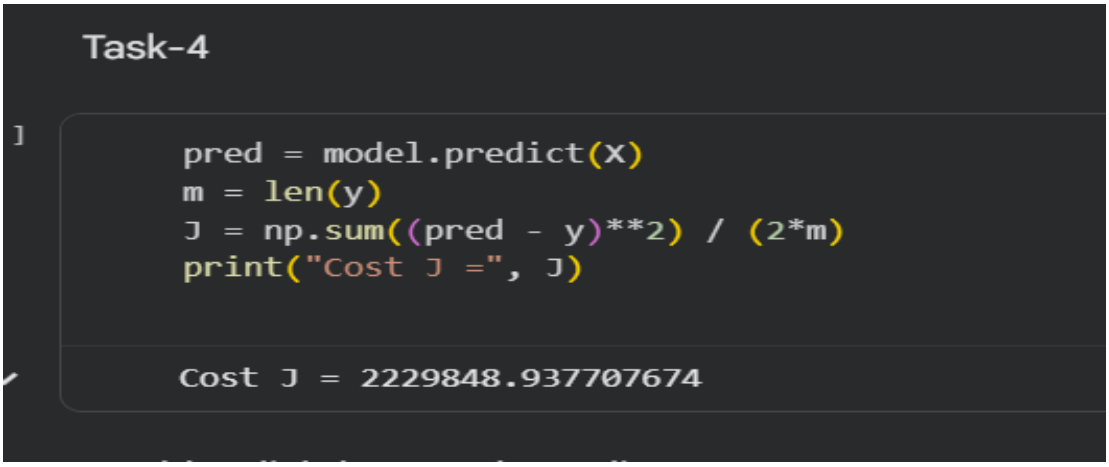
Final answer is Best Model: Sales=23.42×Advertising+167.7

Justification: The model with the least sum of squared errors (LSE) fits the data best since the error between actual sales and predicted sales is the closest to zero for this model.

Task-3(Sales Prediction Using Advertising Budget)

Conclusion: The linear regression model shows a direct positive linear correlation between the sales and the advertising budget. It means that sales are increased with the increased advertising spend and the sales intercept proves that there are still some sales that happen without advertising.

Task-4(Cost Function J)



Conclusion: The cost value gives a measure of the average squared prediction error of the regression model. This value is utilized to assess the extent to which the fitted line represents the data accurately.

PS: this is my link to google collab if needed

(https://colab.research.google.com/drive/1rASQbLtaC5ET10TvvAah_t82sbs5w6Gj?usp=sharing)