**Lab Wk5**

1. Explore regularization through MNIST code:

Regularization is a technique used to prevent overfitting in machine learning models, including logistic regression. In logistic regression, regularization adds a penalty term to the cost function, which discourages the model from fitting the training data too closely. Regularization helps improve the model's generalization performance on unseen data.

In the context of the MNIST dataset and logistic regression, you can use two common types of regularization: L1 regularization (Lasso) and L2 regularization (Ridge). Here's how you can apply regularization to logistic regression on the MNIST dataset using scikit-learn:

import numpy as np

from sklearn.model\_selection import cross\_val\_score

from sklearn.linear\_model import LogisticRegression

from sklearn.datasets import fetch\_openml

# Load the MNIST dataset

mnist = fetch\_openml('mnist\_784')

X = mnist.data

y = mnist.target

# Initialize the logistic regression model with L1 regularization – explain what this call is doing?

logreg\_l1 = LogisticRegression(penalty='l1', solver='saga', max\_iter=100, multi\_class='ovr')

# Perform cross-validation with L1 regularization

num\_folds = 5 # Number of folds for cross-validation

scores\_l1 = cross\_val\_score(logreg\_l1, X, y, cv=num\_folds, scoring='accuracy')

# Print the average accuracy with L1 regularization

average\_accuracy\_l1 = np.mean(scores\_l1)

print(f"Average Accuracy with L1 regularization: {average\_accuracy\_l1:.4f}")

# Initialize the logistic regression model with L2 regularization – explain what this call is doing?

logreg\_l2 = LogisticRegression(penalty='l2', solver='lbfgs', max\_iter=100, multi\_class='ovr')

# Perform cross-validation with L2 regularization

scores\_l2 = cross\_val\_score(logreg\_l2, X, y, cv=num\_folds, scoring='accuracy')

# Print the average accuracy with L2 regularization

average\_accuracy\_l2 = np.mean(scores\_l2)

print(f"Average Accuracy with L2 regularization: {average\_accuracy\_l2:.4f}")

In this example, we use logistic regression with both L1 and L2 regularization. The penalty parameter is set to 'l1' for L1 regularization and 'l2' for L2 regularization. The solver parameter and other hyperparameters can be adjusted based on your needs.

Regularization helps prevent overfitting by shrinking the coefficients of less important features toward zero. L1 regularization can also lead to feature selection, as it encourages some coefficients to become exactly zero.

Remember that hyperparameter tuning is important when using regularization. You can explore different values for parameters like C (inverse of regularization strength) to find the optimal balance between bias and variance for your model.

1. Explore the attached code for Diabetes dataset analysis, optimisation using hyper-parameters and prediction. Walk through the code and figure out what each section is doing as per the following steps. The code is given as Week5Lab-DiabetesPR.zip.

**1. Data Exploration and Preprocessing:**

* Load the Diabetes dataset using a suitable Python library and display the first few rows of the data.
* How many rows and columns does the dataset have?
* Identify the features (attributes) in the dataset. What do these features represent?
* Are there any missing values in the dataset? If so, how could you handle them?
* Calculate basic statistics for the features, such as mean, standard deviation, minimum, and maximum values.

**2. Data Visualization:**

* Create a histogram to visualize the distribution of the target variable ("Outcome").
* Plot a pairplot or scatter matrix to explore relationships between different pairs of features.
* Create a bar plot to visualize the distribution of the "Outcome" variable across different classes.
* Plot a correlation matrix heatmap to visualize the correlation between features. Which features have strong correlations?

**3. Feature Selection:**

* Perform feature selection using a suitable method (e.g., correlation-based or model-based). Which features are most important for predicting the target variable?
* Compare the performance of the model using the selected features versus using all features. Which approach yields better results?

**4. Model Building and Evaluation:**

* Split the dataset into training and testing sets. What's the ratio you chose, and why?
* Train a logistic regression model to predict diabetes outcomes. Evaluate its performance using appropriate metrics (accuracy, precision, recall, F1-score).
* Train a different classification algorithm (e.g., Random Forest, Support Vector Machine) and compare its performance with the logistic regression model.

**5. Model Tuning:**

* Perform hyperparameter tuning for the classification algorithm with the best performance. Use techniques like grid search or random search to find optimal hyperparameters.
* How do the tuned hyperparameters impact the model's performance?

**6. Model Interpretation:**

* Interpret the coefficients of the logistic regression model. Which features have a significant impact on the outcome?
* Use techniques like feature importance plots for other models to understand which features are most influential.

**7. Imbalanced Data:**

* Check if the dataset is imbalanced in terms of the target variable. If so, apply techniques like oversampling or undersampling to balance the classes.
* Evaluate how balancing the dataset affects model performance.

**8. Cross-Validation:**

* Implement k-fold cross-validation to assess the model's performance more robustly. How does the average performance across folds compare to the performance on the test set?