**Assignment 1: Assignment on Python Basics for Machine Learning**

1. Create a Pandas DataFrame containing data of 5 students: their names, ages, and scores in three subjects. Compute the average score for each student and add it as a new column. Filter the DataFrame to show only students with average scores greater than 70.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Age** | **Mathematics** | **Science** | **English** |
| Aarav | 20 | 85 | 78 | 92 |
| Neha | 21 | 65 | 70 | 68 |
| Rohan | 19 | 75 | 82 | 88 |
| Meena | 22 | 55 | 60 | 58 |
| Priya | 20 | 90 | 87 | 93 |

1. Locate and display the values of a column named “Avg\_Score”. Display the data type of this column. Check if there are any missing values in this column.
2. Plot a **scatter plot** where the x-axis represents **ages** and the y-axis represents the **average scores** of the students.
3. Using the above DataFrame,
4. create a bar chart showing the average scores of the students.
5. Create a pie chart showing the distribution of students’ ages.
6. Sort the Dataframe according to the descending order of the Average score.
7. Display the summary statistics (mean, median, std, min, max) of all the numerical columns.
8. Calculate the total marks of each student using apply() and add it as a new column, ‘total\_score’.
9. Add a new column grade based on the ‘total\_score’
10. Plot a histogram of the average scores.
11. Plot a boxplot of all three subject scores to visualize the spread and detect outliers.

**Assignment 2: Assignment on Data Pre-processing – 1**

1. **Dataset**: Titanic dataset
2. Load the Titanic dataset and display the first 5 rows.
3. Check for missing values and report which columns have them.
4. Replace missing values in age column with median.

**2) Dataset**: Titanic dataset

1. Encode the ‘Gender’ column using Label Encoding.
2. Encode ‘Embarked’ column using OneHot encoding.
3. Appy feature scaling on the ‘Fare’ column using StandardScaler
4. Divide data into dependent and independent features and split the dataset into training (80%) and testing (20%) sets.

**Assignment 5: Implementation of Simple Linear Regression – 2**

1. **Dataset: Placement.csv**
2. Load the dataset and display the first 5 rows.
3. Visualize the distribution of the data using a **Scatter Plot.**
4. Prepare the data for training by splitting the data in to training and testing set.
5. Train a **linear regression** model
6. Display the values for the **intercept** and the **slope** of the regression line generated by the model.
7. Evaluate model’s performance using **MSE, MAE, RMSE.**
8. Plot the regression line.
9. Predict the package values for the test set and for student with CGPA 8.5

**Assignment 3: Data Pre-processing – 2**

1. **Dataset:** Students Performance Dataset
2. Simulate missing values: randomly assign NaN to few of the rows in 'math score'. Replace the missing 'math score' values with the mean
3. Apply Label Encoding to the 'gender' and 'lunch' columns
4. Apply **One-Hot Encoding** to 'test preparation course'. Display the shape of the DataFrame before and after encoding.
5. Normalize the 'math score', 'reading score', and 'writing score' using Min-Max Scaling
6. Plot a bar chart showing average scores by gender

**Assignment 4: Implementation of Simple Linear Regression – 1**

1. **Dataset:** Prepare the dataset from a python dictionary and convert it into a Pandas DataFrame.

|  |  |
| --- | --- |
| Hours | Score |
| 1.5 | 20 |
| 2.0 | 25 |
| 2.5 | 30 |
| 3.0 | 35 |
| 3.5 | 45 |
| 4.0 | 50 |
| 4.5 | 55 |
| 5.0 | 60 |
| 5.5 | 65 |
| 6.0 | 70 |

1. Visualize the distribution of the data using a **Scatter Plot.**
2. Prepare the data for training by splitting the data in to training and testing set.
3. Train a **linear regression** model
4. Display the values for the **intercept** and the **slope** of the regression line generated by the model.
5. Evaluate model’s performance using **MSE, MAE, RMSE.**
6. Plot the regression line.
7. Predict the **score** of the student when the hour mentioned is **7.5** Hrs.

**Assignment 6: Implementation of Multiple Linear Regression – 1**

1. **Dataset: Housing.csv**
2. Convert all categorical variables to numeric using appropriate encoding.
3. Divide the data into set of independent variables and dependent variable then Split into training (80%) and testing (20%) sets.
4. Train a Multiple Linear Regression model on the training data.
5. Display the learned coefficients and intercept.
6. Predict prices on the test set. Evaluate using: R² Score Mean Absolute Error (MAE) Mean Squared Error (MSE)

**Assignment 7: Implementation of Multiple Linear Regression – 2**  
**Dataset: Electricity\_cost\_dataset.csv**

1. Check for missing values. If any, handle them appropriately.
2. Check the unique values and convert categorical variables (structure type) using **Label Encoding.**
3. Find average electricity cost for each structure type.
4. Visualization

a) Plot the distribution of the electricity cost.

b) Scatter plot of water consumption vs electricity cost

c) Use a heatmap to visualize correlations among all numerical features.

1. Define your feature matrix X and target variable y. Split the data into training and test sets .
2. Train a Multiple Linear Regression model. Display the model's coefficients and intercept.
3. Calculate the following metrics on the test data:

a) Mean Absolute Error (MAE)

b) Mean Squared Error (MSE)

c) R-squared score

d) Adjusted R-squared score

1. Feature Engineering
2. Create a new feature: water consumption per resident.
3. Train the model again using this feature. Check the performance of model using various evaluation metrics.
4. Plot Residual vs Fitted values to check homoscedasticity.

**Assignment 8: Implementation of Polynomial Regression**

Dataset: SleepQualityPolyRegressionData.csv

1. Load the dataset and plot the histogram (bin=20) to check distributions of Screen\_Time\_Hours and Sleep\_Quality\_Score.
2. Create a scatter plot: Screen\_Time\_Hours vs Sleep\_Quality\_Score. Describe the trend.
3. Train a simple linear regression model. Evaluate with R², MAE, and MSE. Plot the line.
4. Plot Simple Linear Regression line
5. Generate polynomial features (degree 2). Show the resulting feature matrix. Train a polynomial model and evaluate it. Plot the curve over the data.
6. Plot Polynomial regression line
7. Use IQR to find outliers in Sleep\_Quality\_Score. Print total number of outliers.
8. Cap the outliers and retrain polynomial model. Compare metrics before replacing the outliers vs after.
9. Print the intercept and the coefficients of each polynomial term

**Assignment 9: Implementation of K - Nearest Neighbors (KNN) classification – 1** **Dataset: iris.csv**

1. **Data Preprocessing**
2. Load the dataset and display the first 5 rows. Check for missing values.
3. Simulate missing data randomly set 10% of entries in each column to NaN. Replace missing values with the mean.
4. Encode the class labels (Species) as integers.
5. **EDA**
6. Plot a scatter plot of SepalLength vs. PetalLength. Color the points based on their species.
7. **Implement KNN (Using scikit-learn)**
8. Divide the data into features and target. Split the data into 80% training and 20% testing.
9. Use KNeighborsClassifier with: K = 1, 3, 5 and Default distance = Euclidean
10. Train the model and make predictions on the test set.
11. Print predictions alongside actual species for inspection.
12. **Print the confusion matrix.**
13. Use classification\_report to show: Accuracy, Precision, Recall, F1-Score
14. For each value of K used, report the above metrics.

**Assignment 10: Implementation of K - Nearest Neighbors (KNN) classification – 2**

**Dataset:** Titanic (Seaborn library)

1. Check for missing values in the dataset and mention which columns contain them.  
    If any, handle them appropriately using imputation or removal.
2. Drop unnecessary columns that are not useful for modeling (e.g., ‘deck’, ‘embark\_town’, ‘alive’).
3. Encode categorical variables like sex and embarked.
4. Perform Exploratory Data Analysis (EDA):  
    a) Plot the survival count using a bar chart.  
    b) Visualize survival by gender using a countplot.  
    c) Visualize survival by passenger class (pclass).  
    d) Plot the distribution of age with a histogram.
5. Display a heatmap showing correlations among all numerical features.
6. Split the dataset into feature matrix X and target variable y.
7. Scale the features using StandardScaler.
8. Split the data into training and testing sets using train\_test\_split.
9. Cretae a error plot to find optimal value of K. (use range of K between 1-5).
10. Train a KNN classifier using the optimal K value
11. Evaluate the model using the following metrics (Use optimal K Value):  
     a) Confusion Matrix  
     b) Classification Report  
     c) Accuracy Score

**Assignment 11: Implementation of Logistic Regression**

Dataset: Pima\_Indians\_Diabetes.csv

1. Load the Pima Indians Diabetes dataset into a Pandas DataFrame and display the first 5 rows.
2. Check and report the number of missing values in each column of the dataset.
3. Replace any zero values in the 'Glucose', 'BloodPressure', and 'BMI' columns with the median of their respective columns.
4. Generate histograms for the 'Glucose' and 'BMI' features to visualize their distributions.
5. Create a new binary feature called 'High\_BMI' where value 1 indicates BMI > 30 and 0 otherwise.
6. Standardize all numeric feature columns using StandardScaler and show the first 5 rows after scaling.
7. Split the dataset into train and test sets using an 80:20 ratio with a random state of 42.
8. Train a logistic regression model on the training data using default parameters.
9. Predict the test set outcomes and calculate accuracy, precision, recall, and F1-score.

**Assignment 12: Implementation of Logistic Regression - 2**

1. Load the Breast Cancer Wisconsin dataset from **sklearn.datasets.load\_breast\_cancer** into a Pandas DataFrame. Display the first 5 rows.
2. Print the number of instances for each class in the target variable.
3. Divide data into feature and target. Split the data into training and test sets with a 70:30 ratio, Display the shapes of both sets.
4. Train a logistic regression model using the training set.
5. Evaluate and print the accuracy and recall for each class on the test set, as well as the full classification report.
6. Extract the coefficients from the trained logistic regression model. For each feature, compute the absolute value of its coefficient. List and display the 5 features with the highest absolute coefficient values.
7. Using only the 5 most influential features identified above, retrain a logistic regression model. Evaluate and compare the accuracy and recall with your previous full-feature model.

**Assignment 13: Implementation of Naive Bayes Theorem**

Dataset: spam\_binary\_features.csv

1. Load the dataset and display the first 10 rows. Identify the number of spam and non-spam messages.
2. Create a bar plot showing the count of spam vs. non-spam messages using Seaborn
3. For each of the keywords (free, win, call, etc.), calculate how frequently they appear in spam vs. non-spam messages. Present this in a bar chart (keyword on x-axis, frequency on y-axis, hue = spam/not-spam)
4. Plot a heatmap showing correlations among all keyword features to explore multicollinearity or feature relationships.
5. Split the dataset into training and testing sets in a 70:30 ratio. Print the shape of the resulting sets.
6. GaussianNB

6.1. Train a GaussianNB classifier on the training set

6.2. Evaluate its accuracy, precision, recall, and F1 score. Display the confusion matrix.

1. MultinomialNB

7.1. Train a MultinomialNB classifier on the training set

7.2. Evaluate its accuracy, precision, recall, and F1 score. Display the confusion matrix. Compare the performance metrics of both classifiers.

1. After training, print the class prior probabilities learned by both the Naive Bayes models.
2. Use a horizontal bar chart to show the top 5 most indicative features of spam based on conditional probabilities.
3. Manually create a few test samples (binary vectors of keyword presence) and predict whether each message is spam or not using the trained model.

**Assignment 14: Implementation of Support Vector Machine**

Dataset: Crop Recommendation Dataset

1. Load the dataset.
   1. Display the first 5 rows
   2. Show the shape and column names
   3. Check unique values in the target variable (label)
   4. Check for any missing or null values in the dataset.
   5. Display data types and basic statistics of the features.
2. Count the number of records for each crop and display the counts as a table.
3. Plot histograms or boxplots for:
   1. N, P, K
   2. temperature, humidity, ph, rainfall
4. Encode the label (crop name) into numeric format using LabelEncoder.
5. Show correlation between features using a heatmap.
6. Divide the data into features and label. Split the dataset into training (80%) and testing (20%) sets using train\_test\_split.
7. Train an SVM classifier using the **linear kernel**.
8. Print classification report and confusion matrix. Report accuracy score
9. Train another SVM classifier using the **RBF kernel**. Print classification report. Compare accuracy and F1-score with the linear kernel model.
10. Create a function recommend\_crop() that takes input values for N, P, K, temperature, humidity, ph, and rainfall, and returns the predicted crop name.

**Assignment 15: Implementation of Decision Tree Classification**

**Dataset:** Penguins dataset from Seaborn (sns.load\_dataset('penguins'))

1. Load the penguins dataset. Display:
2. First 10 rows
3. Shape of the dataset
4. Column names and data types
5. Identify missing values. Fill missing **numerical** values with the mean and **categorical** values with the mode.
6. Plot a **countplot** showing the number of penguins per species.
7. Convert categorical features (island, sex) into numeric form using Label Encoding.
8. Use the features: bill\_length\_mm, bill\_depth\_mm, flipper\_length\_mm, body\_mass\_g, sex and Target: species. Split into **80% training** and **20% testing** set
9. Train a **Decision Tree Classifier** with criterion="entropy". Print:
10. Accuracy on training set
11. Accuracy on testing set
12. Tree depth
13. Number of leaves
14. Plot the trained decision tree using plot\_tree() with proper feature and class names.
15. Train a Decision Tree Classifier with criterion="gini". Print the same metrics as in Question 6.
16. Plot the trained decision tree for the Gini index model.
17. **Compare Entropy vs Gini:**  Create a small comparison table for:
18. Criterion used
19. Training accuracy
20. Testing accuracy
21. Tree depth

**Assignment 16: Implementation of K-Means Clustering – 1**

**Dataset: Mall\_Customer.csv**

1. Load the dataset using pandas and display the first 5 rows using .head().
2. Display dataset information, shape, and check for any missing values.
3. Drop the ID column (e.g., CustomerID or ID) as it is not a meaningful feature for clustering.
4. Generate summary statistics using .describe() and visualize the relationships using a pairplot.
5. Encode the Gender column. (e.g., Male = 0, Female = 1).
6. Select only the numerical features and perform feature scaling using StandardScaler.
7. Determine the optimal number of clusters (k) using the Elbow Method: Loop through k = 2 to 10 and Record inertia\_ for each model and Plot the Elbow graph (k vs inertia)
8. Based on the Elbow graph and visual separation, choose the best value for k.
9. Train the final KMeans model using the chosen number of clusters and assign cluster labels to the DataFrame.
10. Visualize the clusters using a scatter plot of the first two features (e.g., Age vs Spending Score).

**Assignment 17: Implementation of K-Means Clustering – 2**

Dataset: Spotify songs dataset

1. Load the dataset and display the first 10 rows. Show dataset shape and column names.
2. Select only **numeric audio features** (drop track/artist names, release year, genre).
3. Check for missing values and handle them (drop/mean impute).
4. Scale all selected features using StandardScaler.
5. Run **K-Means for k = 2 to 10**, compute the **Silhouette Score** for each k, and plot **k vs. Silhouette Score**.
6. Based on the Silhouette plot, select the best k value.
7. Fit K-Means with the chosen k and assign cluster labels to each song.
8. Compute the **average values of each audio feature per cluster** and show in a table.
9. Visualize clusters in **2D scatter plots** of interesting feature pairs:
   * Danceability vs. Energy
   * Acousticness vs. Instrumentalness  
      (color points by cluster).

**Assignment 18: Implementation of Agglomerative Clustering**

Q.1. **Dataset: Iris.csv**

1. Load the Iris dataset using *sklearn.datasets.load\_iris.* Print the shape of data and target. Display the first 5 rows of features.
2. Apply Agglomerative Clustering with n\_clusters=3 (default linkage). Print cluster labels. Show how many samples are in each cluster.
3. Reduce the Iris dataset to 2D using PCA.
4. Plot the 2D PCA-reduced dataset with colors based on Agglomerative Clustering labels.

Q.2. Dataset: Mall\_Customers.csv

1. Load the Mall\_Customers.csv dataset using pandas. Print the shape of data. Display the first 5 rows.
2. Select the columns Annual Income (k$) and Spending Score (1–100) for clustering. Plot a scatter plot of these two features before clustering.
3. Apply Agglomerative Clustering with n\_clusters=5 (default linkage). Print cluster labels. Show how many samples are in each cluster.
4. Apply Agglomerative Clustering with the following linkages on the dataset: single, complete, average, ward
5. For each method, Plot the 2D scatter plot (Annual Income vs Spending Score) with colors based on cluster labels.

**Assignment 19: Implementation of Apriori**

Dataset: Market\_Basket\_Optimisation.csv

Q.1. Load dataset using pandas. Display the shape of the dataset and first 5 transactions.

Q.2. Convert the dataset into a **list of transactions**, where each transaction is a list of purchased items. Print first 3 transactions.

Q.3. Use **TransactionEncoder** from mlxtend.preprocessing to encode transactions into a binary format (0/1). Print the shape of the encoded dataset and the first 5 rows.

Q.4. Apply the **Apriori algorithm** with min\_support=0.02. Display the top 10 frequent itemsets sorted by support.

Q.5. Run Apriori again with min\_support=0.05. Compare results.

Q.6. Use association\_rules with metric="confidence" and min\_threshold=0.05. Display the top 10 rules with support, confidence, and lift.

Q.7. Filter rules with lift > 2. Print top 5 rules.

Q.8. Plot a **scatter plot** of rules:

* X-axis = Support
* Y-axis = Confidence
* Color (or size) = Lift

**Assignment 20: Implementation of FP Growth**Dataset: Market\_Basket\_Optimisation.csv

1. 1. Load dataset & print first 10 transactions.
2. Convert dataset into a list of lists & display 5 transactions.
3. Convert transactions into one-hot encoded format using TransactionEncoder().
4. Apply FP-Growth to mine frequent itemsets
5. Sort & Display Top 10 Frequent Itemsets
6. Using the frequent itemsets obtained from FP-Growth, generate association rules using mlxtend.frequent\_patterns.association\_rules. (Set minimum confidence threshold = 0.5.)