**Theory of Each FML Practical**

Exp 0: -

1. Import the required libraries like Pandas, Numpy and Matplotlib and read the data using pd.read\_csv().
2. Use iloc to assign values to x and y.
3. Impute missing values using SimpleImputer from sklearn.impute and fit(), transform().
4. Apply One Hot Encoding (OHE) for x and Label Encoding for y.
5. Apply train\_test\_split from sklearn to create x\_train, y\_train, x\_test and y\_test.

Exp 1: -

1. Import the required libraries like Pandas, Numpy and Matplotlib and read the data using pd.read\_csv().
2. Print the first five values using df.head() and use iloc to assign values to x and y.
3. Apply train\_test\_split after importing it from sklearn.
4. Import LinearRegression from sklearn.linear\_model and create an object named regressor to implement Linear Regression, using regressor.fit(x\_train,y\_train).
5. Find y\_pred using regressor.predict(x\_test).
6. Using Matplotlib.pyplot’s scatter and line plot, plot the regression line on a graph.
7. Predict salary for any random amount of experience and print as the output.

Dataset: -

The dataset used is a simple 30 row and 2 column dataset containing Years of Experience(independent variable) and Salary(dependant variable) of some employees. It has no missing values and no categorical features.

Exp 2: -

1. Logistic Regression is a statistical method used for binary classification, i.e. it predicts the probability of the data point to belong to 2 possible cases (0 or 1). It uses the sigmoid function to model the output.
2. Import the required libraries like Pandas, Numpy and Matplotlib and read the data using pd.read\_csv().
3. Print the first five values using df.head() and use iloc to assign values to x and y.
4. Apply train\_test\_split after importing it from sklearn.
5. Apply Feature Scaling by importing StandardScaler class from sklearn.preprocessing and fit\_transform(x\_train) and transform(x\_test)
6. Import LogisticRegression from sklearn.linear\_model and create an object named classifier to implement Logistic Regression, using classifier.fit(x\_train,y\_train).
7. Find y\_pred using classifier.predict(x\_test) and print the records vertically using reshape and concatenate functions.
8. Predict if the product was purchased for a few random values of age and estimated salary, printing the result as the output.

Dataset: -

The dataset contains 400 rows and 3 columns of age, estimated salary and purchased respectively, all having numerical values. There are no missing data points, eliminating the need for data imputation. The first 2 columns (Age and Estimated Salary) form the independent variable while purchased (having values between 0 and 1) is the dependant variable to be predicted.

Exp 3 (Beyond Curriculum): -

1. Multiple linear regression is a regression model that estimates the relationship between a quantitative dependent variable and two or more independent variables using a straight line. (y^= bo + blX1 + b2X2 + ...+ bnXn)
2. Import the required libraries like Pandas, Numpy and Matplotlib and read the data using pd.read\_csv().
3. Print the first five values using df.head() and use iloc to assign values to x and y.
4. Apply OHE (One Hot Encoding) for values of x and then train\_test\_split.
5. Import LinearRegression from sklearn.linearmodel and pass values of x\_train and y\_train to fit the model.
6. Find y\_pred and use reshape, concatenate functions to print y\_pred along with y\_test vertically for comparing values.
7. Polynomial Regression is a form of regression analysis in which the relationship between the independent variables and dependent variables are modelled in the nth degree polynomial. Since only the independent variables are raised to degree n and the coefficients are still linear, it is called polynomial regression.
8. Repeat steps 2 and 3 to reach train\_test\_split.
9. Import both LinearRegression model (step 5) and Polynomial Features. Create object poly\_reg for polynomial features and fit\_transform values of x1 to get x\_poly.
10. Now we may visualize results from both methods. Linear Regression gives a straight line that does not fit very accurately but Polynomial Linear Regression’s curve fits better to give more accuracy

Datasets: -

1. The first dataset contains 50 rows and 5 columns with data from 50 startups to predict their profit. We need to apply OHE in the fourth column to convert categorical values of states to numerical values.
2. The second dataset contains 10 rows with information about the expected/predicted salary for a person at a certain position/level in the company/organization. We don’t need to apply OHE as the level column already converts the positions to numerical values, allowing us to skip column 1 and take only column 2 as x values and column 3 with salaries as values of y.

Exp 3:

1. K-Nearest Neighbors (KNN) is a simple way to classify things by looking at what’s nearby where, K is just a number that tells the algorithm how many nearby points (neighbours) to look at when it makes a decision. KNN uses distance metrics to identify nearest neighbour (Euclidean/Manhattan/Minkowski distances). It is also called as a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification it performs an action on the dataset.
2. Import the required libraries like Pandas, Numpy and Matplotlib and read the data using pd.read\_csv().
3. Print the first five values using df.head() and use iloc to assign values to x and y.
4. Apply train\_test\_split after importing it from sklearn.
5. Apply Feature Scaling by importing StandardScaler class from sklearn.preprocessing and fit\_transform(x\_train) and transform(x\_test)
6. Import KNeigborsClassifier from sklearn.neighbors and create an object named classifier to implement KNN with k=5 and Minkowski method with p=2(for Euclidean Distance).
7. Find y\_pred using classifier.predict(x\_test) and print the records vertically using reshape and concatenate functions.
8. Predict if the product was purchased for a random values of age and estimated salary, printing the result as the output. Also generate a Confusion Matrix to display True Positive, True Negative, False Positive and False Negative values

Datasets:

The dataset contains 400 rows and 3 columns of age, estimated salary and purchased respectively, all having numerical values. There are no missing data points, eliminating the need for data imputation. The first 2 columns (Age and Estimated Salary) form the independent variable while purchased (having values between 0 and 1) is the dependant variable to be predicted.

Exp 4:

1. Support Vector Machine (SVM) is a supervised machine learning algorithm particularly well-suited for classification tasks. SVM aims to find the optimal hyperplane in an N-dimensional space to separate data points into different classes. The algorithm maximizes the margin between the closest points of different classes. It forms a decision boundary (Hyperplane) separating different classes in feature space, represented by the equation**wx + b = 0** in linear classification. The **Support Vectors** are the closest data points to the hyperplane, crucial for determining the hyperplane and margin in SVM, wherein the **Margin** represents the distance between the hyperplane and the support vectors. SVM aims to maximize this margin for better classification performance.
2. Import the required libraries like Pandas, Numpy and Matplotlib and read the data using pd.read\_csv().
3. Print the first five values using df.head() and use iloc to assign values to x and y.
4. Apply train\_test\_split after importing it from sklearn.
5. Apply Feature Scaling by importing StandardScaler class from sklearn.preprocessing and fit\_transform(x\_train) and transform(x\_test)
6. Import SupportVectorClassifier (SVC) from sklearn.svm and create an object named classifier to implement SVM with kernel=linear and Random State=0.
7. Find y\_pred using classifier.predict(x\_test) and print the records vertically using reshape and concatenate functions.
8. Predict if the product was purchased for a random value of age and estimated salary, printing the result as the output. Also generate a Confusion Matrix and print the accuracy.

Datasets:

The dataset contains 400 rows and 3 columns of age, estimated salary and purchased respectively, all having numerical values. There are no missing data points, eliminating the need for data imputation. The first 2 columns (Age and Estimated Salary) form the independent variable while purchased (having values between 0 and 1) is the dependant variable to be predicted.

Exp 5:

1. Random Forest is a popular machine learning algorithm that can be used for both Classification and Regression and is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.
2. Bagging comes from the word **B**ootstrap **AGG**regat**ING. Bootstrapping -** Bootstrapping is the process of resampling subsets of data with replacement from the initial dataset. These subsets of data are called bootstrapped subsets. Resampled with replacement means an individual data point can be sampled multiple times. Each bootstrap dataset is used to train a weak learner. **Aggregating -** The individual weak learners are trained independently of each other. Each learner makes independent prediction and in the end they are all aggregated together either by majority voting or averaging
3. Hence Random Forest Classifier contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.
4. Import the required libraries like Pandas, Numpy and Matplotlib and read the data using pd.read\_csv().
5. Print the first five values using df.head() and use iloc to assign values to x and y.
6. Apply train\_test\_split after importing it from sklearn.
7. Apply Feature Scaling by importing StandardScaler class from sklearn.preprocessing and fit\_transform(x\_train) and transform(x\_test)
8. Import RandomForestClassifier from sklearn.ensemble and create an object named classifier to implement Random Forest with 10 estimators and criterion entropy.
9. Find y\_pred using classifier.predict(x\_test) and print the records vertically using reshape and concatenate functions.
10. Predict if the product was purchased for a random value of age and estimated salary, printing the result as the output. Also generate a Confusion Matrix and print the accuracy.

Exp 6:

1. Naive Bayes classifiers are supervised machine learning algorithms used for classification tasks, based on Bayes’ Theorem to find probabilities. It is named as “Naive” because it assumes the presence of one feature does not affect other features. The “Bayes” part of the name refers to the basis in Bayes’ Theorem. In [Gaussian Naive Bayes](https://www.geeksforgeeks.org/gaussian-naive-bayes/), continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution. A Gaussian distribution is also called [Normal distribution](https://en.wikipedia.org/wiki/Normal_distribution) When plotted, it gives a bell shaped curve which is symmetric about the mean of the feature values.
2. Import the required libraries like Pandas, Numpy and Matplotlib and read the data using pd.read\_csv().
3. Print the first five values using df.head() and use iloc to assign values to x and y.
4. Apply train\_test\_split after importing it from sklearn.
5. Apply Feature Scaling by importing StandardScaler class from sklearn.preprocessing and fit\_transform(x\_train) and transform(x\_test)
6. Import GaussianNBfrom sklearn.naive\_bayes and create an object named classifier to implement Naïve Bayes in Gaussian form.
7. Find y\_pred using classifier.predict(x\_test) and print the records vertically using reshape and concatenate functions.
8. Predict if the product was purchased for a random value of age and estimated salary, printing the result as the output. Also generate a Confusion Matrix and print the accuracy.

Datasets:

The dataset contains 400 rows and 3 columns of age, estimated salary and purchased respectively, all having numerical values. There are no missing data points, eliminating the need for data imputation. The first 2 columns (Age and Estimated Salary) form the independent variable while purchased (having values between 0 and 1) is the dependant variable to be predicted.

Exp 7:

1. A decision tree is a graphical representation of different options for solving a problem and show how different factors are related. It has a hierarchical tree structure starts with one main question at the top called a node which further branches out into different possible outcomes where: **Root Node** is the starting point that represents the entire dataset. **Branches**: These are the lines that connect nodes. It shows the flow from one decision to another. **Internal Nodes**are Points where decisions are made based on the input features. **Leaf Nodes**: These are the terminal nodes at the end of branches that represent final outcomes or predictions
2. Import the required libraries like Pandas, Numpy and Matplotlib and read the data using pd.read\_csv().
3. Print the first five values using df.head() and use iloc to assign values to x and y.
4. Apply train\_test\_split after importing it from sklearn.
5. Apply Feature Scaling by importing StandardScaler class from sklearn.preprocessing and fit\_transform(x\_train) and transform(x\_test)
6. Import DecisionTreeClassifier from sklearn.tree and create an object named classifier to implement Decision Trees with criterion=entropy and Random State=0.
7. Find y\_pred using classifier.predict(x\_test) and print the records vertically using reshape and concatenate functions.
8. Predict if the product was purchased for a random value of age and estimated salary, printing the result as the output. Also generate a Confusion Matrix and print the accuracy.

Datasets:

The dataset contains 400 rows and 3 columns of age, estimated salary and purchased respectively, all having numerical values. There are no missing data points, eliminating the need for data imputation. The first 2 columns (Age and Estimated Salary) form the independent variable while purchased (having values between 0 and 1) is the dependant variable to be predicted.

Exp 8:

1. K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabelled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on. It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters. The k-means clustering algorithm mainly performs two tasks:

* Determines the best value for K centre points or centroids by an iterative process.
* Assigns each data point to its closest k-centre. Those data points which are near to the particular k-centre, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

1. Import the required libraries like Pandas, Numpy and Matplotlib and read the data using pd.read\_csv().
2. Use iloc to assign values to x.
3. Import KMeans from sklearn.clusters and create an empty list named wcss. Apply KMeans on first 10 values of x for plotting the elbow curve, with x-axis being number of clusters and y-axis being wcss.
4. Create object named kmeans to implement clustering with clusters = 5 and init=’k-means++’ with Random State=42.
5. Print a graph with 5 different clusters groups to show the 5 different clusters with title as cluster of customers and x-axis being annual income(k$) and y-axis being spending score (1-100).

Dataset: -

The given dataset contains data of 200 customers in the form of 200 rows and 5 columns (CustomerID, Gender, Age, Annual Income(k$) and Spending Score(1-100)) which we will be using to create 5 clusters using the K-Means Clustering. The dataset has no missing values and is free from any sort of errors and repetition which makes it easier for us to directly apply the Clustering.

Project: -

This project implements a Spam Email Detection system using Natural Language Processing and Machine Learning with a user interface built in Streamlit. It begins by loading and cleaning the dataset (spam.csv), removing duplicates and checking for missing values. The labels "ham" and "spam" are replaced with "Not Spam" and "Spam" respectively. The text messages are then split into training and testing sets, and CountVectorizer is used to convert the text data into numerical format by removing stop words. A Multinomial Naive Bayes classifier is trained on the transformed training data. The model’s performance is evaluated using accuracy, a classification report, and a confusion matrix, which is visualized using a heatmap. Additionally, a word cloud is generated to highlight the most frequent words in the messages. Finally, a Streamlit web interface allows users to input their own message and get real-time predictions indicating whether the message is spam or not.