Approaches:

1)KNN,SVM,LogReg:

Common Steps for All Models

- Load the wine dataset.
- Explore the data (head, tail, nulls, stats).
- Plot feature distributions and a correlation heatmap.
- Split data into features (X) and target (quality).
- Standardize the features for better model performance.
- Split into training and testing sets.

K-Nearest Neighbors (KNN)

- Compares each test sample with nearby training samples.
- Tries different values of k (3, 5, 7, etc.) using grid search.
- Picks the best k, trains the model, and checks accuracy.

Support Vector Machine (SVM)

- Finds a boundary that separates classes best.
- Tests different settings for kernel and regularization (C).
- Chooses the best combo, trains the model, and evaluates it.

Logistic Regression

- Estimates the probability of each class.
- Tries different regularization strengths (C) via grid search.
- Picks the best, trains the model, and checks performance.

2)Desctree,Linreg,Xgboost

Common Steps

- Load stock data from CSV and sort it by date.
- Extract Day, Month, and Year from the Date.
- Plot stock prices over time and show feature correlations.
- Split data into input features (like Open, High, Volume, etc.) and target (Close price).
- Split into training and testing sets (no shuffling for time series).

Decision Tree Regressor

- Learns decision rules from the data.
- Fits a tree that splits the data based on feature values.
- Good for quick insights but may overfit.

XGBoost Regressor

- Uses boosted decision trees for high accuracy.
- Learns from errors of previous trees (gradient boosting).
- Often gives the best performance in structured data.

Linear Regression

- Tries to fit a straight line (or hyperplane) through the data.
- Assumes a linear relationship between features and target.
- Simple and interpretable, but may underperform with complex data.

Model Comparison

Each model is evaluated using:

RMSE (Root Mean Squared Error) – measures prediction error.

R^2 Score – shows how well the model explains the variance in target.

3)Random Forest:

Loaded Data:
Loaded both the training (train.csv) and test (test.csv) datasets.

Data Preview:
Printed the first and last few rows, basic stats, and missing value counts.

Data Cleaning:

- Filled missing Age and Fare values with the median.
- Removed irrelevant columns like Name, Ticket, Cabin, and Embarked.

Encoding:
Converted categorical text (like Sex) into numbers using Label Encoding.

Visualization (EDA):

- Heatmap to show feature correlation.
- Pair plots, count plots, and histograms to explore patterns (e.g., survival by age, sex, class).

Feature Setup:

- Chose Survived as the target (what to predict).
- Dropped PassengerId and separated features for training and testing.

Feature Scaling:
Used StandardScaler to normalize features.

Train/Test Split:
Split the data (80% train, 20% test) while preserving class balance (stratify).

Model Training:
Trained a **Random Forest** with controlled depth and split rules to prevent overfitting.

Predictions + Evaluation:

- Made predictions on training and test sets.
- Printed accuracy scores and classification report.
- Plotted a confusion matrix to visualize correct vs wrong predictions.

Final Output:

- Predicted survival for the test set.
- Saved results to result_new.csv for submission or further use.

4)DBSCAN,Kmeans:

1. Data Preparation & EDA

- Loads and merges List of Orders.csv and Order Details.csv.
- Displays data insights and handles nulls.
- Visualizes correlations with heatmap and pairplots.

2. Preprocessing

- Selects ['Quantity', 'Profit', 'Amount'] as clustering features.
- Applies **log transformation on reflected data** to reduce skewness.
- Scales using StandardScaler.
- Reduces dimensionality to 2D using **PCA** for visualization.

3. Clustering Algorithms

- **DBSCAN**: Density-based clustering.
- **KMeans**: Centroid-based clustering.
- **KMedoids**: Similar to KMeans but uses actual data points as cluster centers.
- Plots the clusters in 2D PCA space and prints **silhouette scores** and **cluster distributions**.