30 ML TASKS

This document presents a comprehensive compilation of 30 machine learning (ML) This document contains 30 machine learning tasks completed as part of a learning program by GeakMinds, under the guidance of Mr. Bishwanath Roy. The tasks cover various ML techniques, including classification, regression, and clustering, using real-world datasets. Each task focuses on applying core concepts, building models, and drawing insights, helping to strengthen practical machine learning skills.

Krishna Chaitanya Muttevi

TASK 1 - Linear Regression from Scratch

DATASET - https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_regression.html

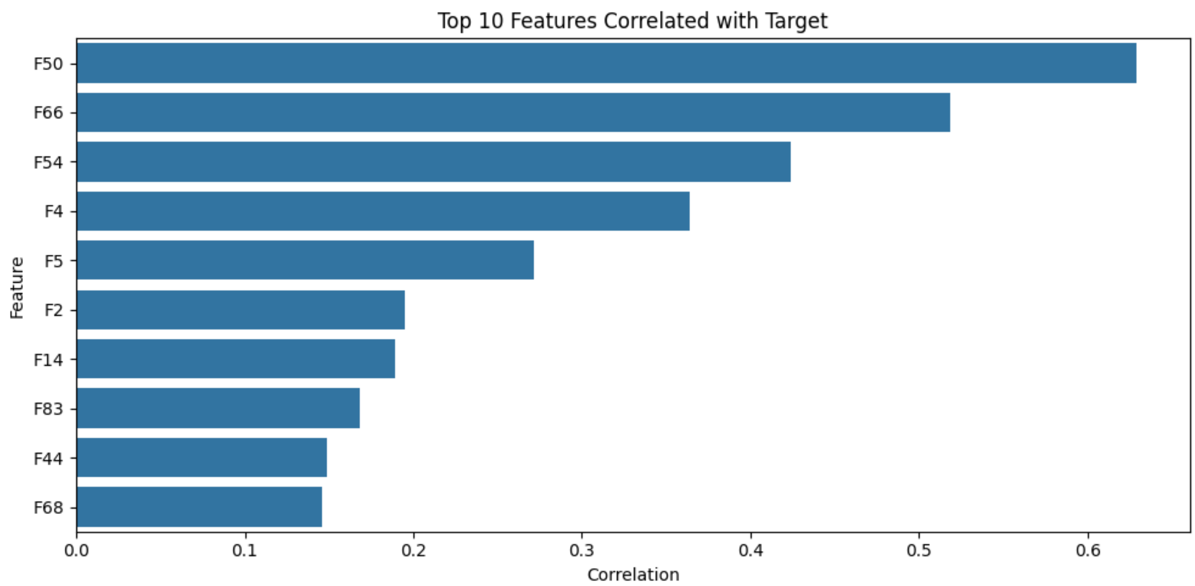
AlGORITHM USED : MULTIPLE LINEAR REGRESSION

WORKING :

1. EXPLORED THE DATASET, PERFORMED CORRELATION AMONG FEATURES
2. SPLITTED THE DATA INTO TRAINING AND TESTING
3. APPLIED THE REGRESSION BASED ON EQUATION

   TARGET VAR = INTERCEPT + SLOPE * DEPENDENT VAR
4. COMPARED WITH THE SKLEARN'S LINEAR REGRESSION

RESULTS :



CUSTOM DEFINED REGRESSION:

```
Train MSE: 1.1636774692626083e-25
Test MSE : 4323.549282609869
Test R²  : 0.7336747955896765
```

SKLEARN'S REGRESSION :

```
Train MSE: 8.093658715010105e-26
Test MSE : 4342.431179766957
Test R²  : 0.7325116944449197
```

TASK 2 - Spam Classifier with Naive Bayes

DATASET - https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset

AlGORITHM USED : MultinomialNB

WORKING :

1. THE DATASET HAS 2 CATEGORIES HAM , SPAM
2. PREPROCESSED THE DATA BY REMOVING NULL , NON ALPHABETS , LOWERED THE TEXTS, LEMMATIZED
3. SPLITED THE DATA , APPLIED MULTI NOMIAL NAIVE BAYES BASED ON THE BAYES THEROM'S CONDITIONAL PROBABILITY

   P(C/x) = (P(C)* P(x/ C)) / P(x)

4. MULTINOMIALNB ASSUMES WORD OCCUR INDEPENDENTLY

$$P(x \mid C_k) = \prod_{i=1}^{n} P(w_i \mid C_k)^{x_i}$$

5. **AFTER I HAVE TUNED IT WITH GRIDSEARCH CV FOR FINDING BEST ESTIMATOR**

RESULTS:

	Class	Text
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

ACCURACY BEFORE TUNING : 0.9802513464991023

BEST PARAMS : {'alpha': 0.1}

BEST ESTIMATOR SCORE : 0.9910273665320771

TASK 3 - TITANIC SURVIVAL PREDICTION

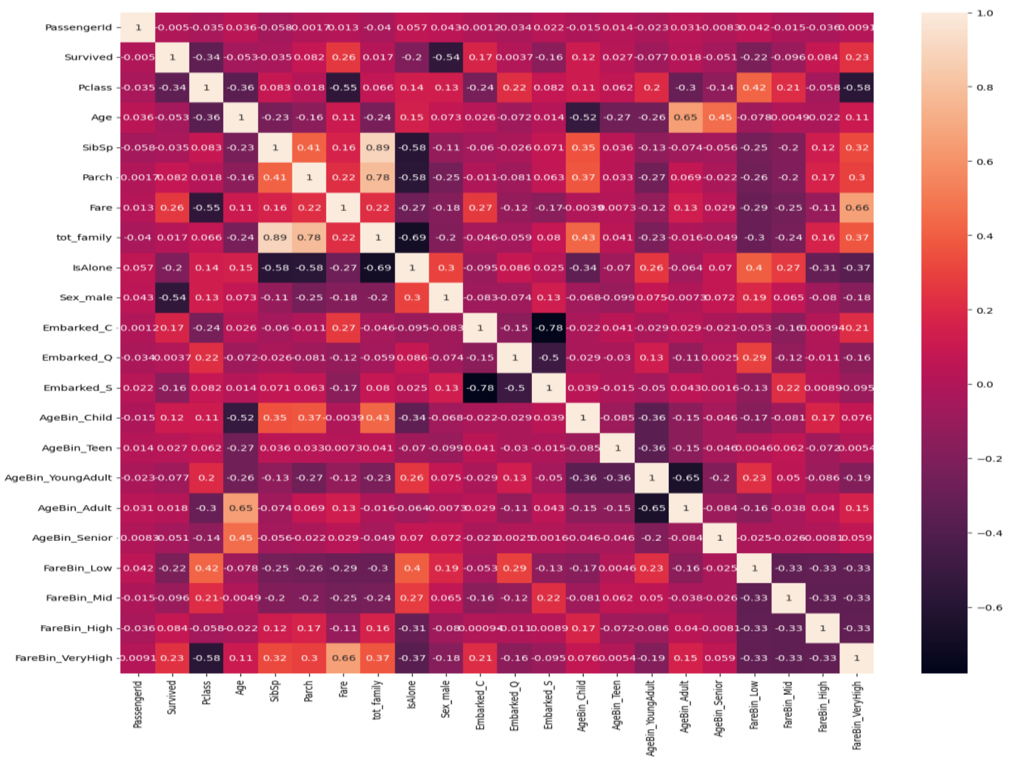
DATASET - https://www.kaggle.com/c/titanic/data

AlGORITHM USED : XGBOOST CLASSIFIER (SUPERVISED)

WORKING :

1. Loaded the Titanic dataset containing passenger details like age, fare, class, and survival status.
2. Filled missing values in the Age column using its mode.
3. Created new features: tot_family (family size), and IsAlone (if the passenger traveled alone).
4. Binned Age into categories (Child, Teen, YoungAdult, etc.) and Fare into quantiles.
5. Applied one-hot encoding to categorical features including Sex, Embarked, AgeBin, and FareBin.
6. Dropped irrelevant columns like Name, Ticket, and Cabin.
7. Initialized and trained an XGBoost Classifier on the processed data.
8. Evaluated model performance using accuracy, precision, recall, and F1-score.
9. Compared performance with baseline models to check improvement using boosting.

RESULTS :



ACCURACY : 84.688995215311

TASK 4 - CUSTOMER SEGMENTATION USING KMEANS CLUSTERING

DATASET - https://www.kaggle.com/datasets/vjchoudhary7/customer-segmentation-tutorial

AlGORITHM USED : KMEANS CLUSTERING (UNSUPERVISED)

WORKING :

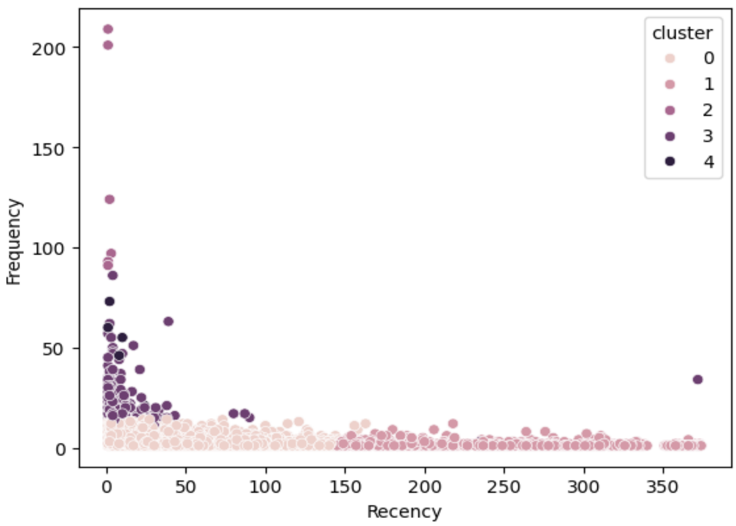
1. THE DATASET CONTAINS CUSTOMER INFORMATION INCLUDING FEATURES LIKE AGE, ANNUAL INCOME, AND SPENDING SCORE
2. THE DATASET WAS TRANSFORMED INTO RFM FORMAT

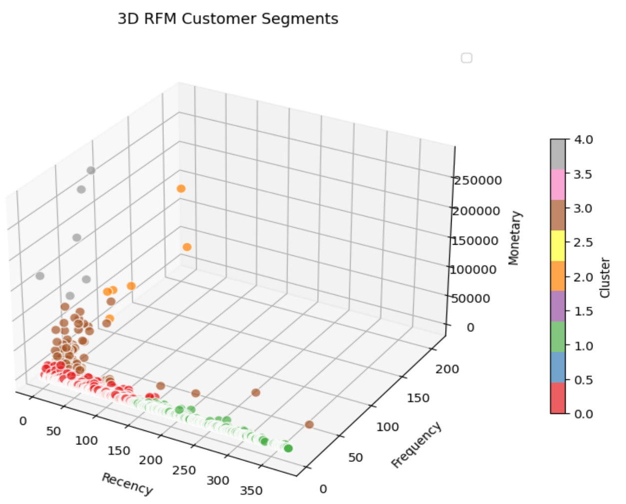
   RECENCY: DAYS SINCE LAST PURCHASE

   FREQUENCY: TOTAL NUMBER OF PURCHASES

   MONETARY: TOTAL MONEY SPENT
3. PREPROCESSED THE DATA BY HANDLING MISSING VALUES (IF ANY), SELECTING RELEVANT FEATURES (E.G., ANNUAL INCOME AND SPENDING SCORE), AND SCALING IF REQUIRED
4. USED THE ELBOW METHOD TO FIND THE OPTIMAL NUMBER OF CLUSTERS (K) BY PLOTTING WITHIN-CLUSTER-SUM-OF-SQUARES (WCSS)
5. USED THE ELBOW METHOD TO FIND THE OPTIMAL NUMBER OF CLUSTERS (K) BY PLOTTING WITHIN-CLUSTER-SUM-OF-SQUARES (WCSS)
6. APPLIED KMEANS CLUSTERING TO GROUP CUSTOMERS INTO K CLUSTERS BASED ON SIMILAR SPENDING BEHAVIOR
7. KMEANS MINIMIZES THE VARIANCE WITHIN EACH CLUSTER AND ASSUMES CLUSTERS ARE SPHERICAL AND EQUAL IN SIZE

RESULTS:





K (ELBOW METHOD) : 5

SILHOUETTE SCORE: 0.62

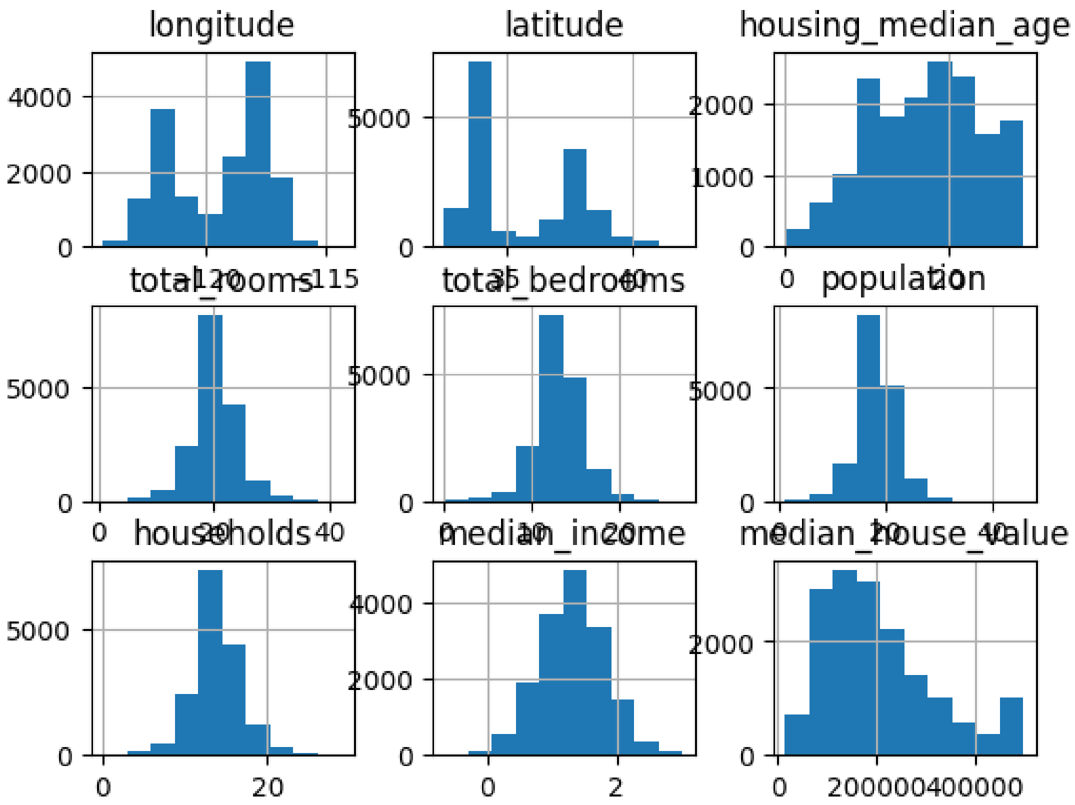
TASK 5 - Random Forest House Price Model

DATASET - https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data

AlGORITHM USED : RANDOM-FOREST REGRESSOR

WORKING :

1. Loaded the California housing dataset containing features like income, rooms, population, etc.
2. Merged x_test and y_test into a single DataFrame for preprocessing.
3. Applied Box-Cox transformation on skewed numeric columns like total_rooms, households, and median_income.
4. Used one-hot encoding to convert the categorical feature ocean_proximity into numerical format.
5. Created a new feature bhk_per_household = total_bedrooms / households, and applied log transformation to reduce skew
6. Applied log transformation to the population column.
7. Trained a baseline Random Forest Regressor using default parameters.
8. Tuned model using RandomizedSearchCV with cross-validation to find the best hyperparameters (like n_estimators, max_depth.



BEST ESTIMATOR SCORE : 79.62236962187191

TASK 6 – IRIS FLOWER CLASSIFICATION USING SVM (RBF KERNEL)

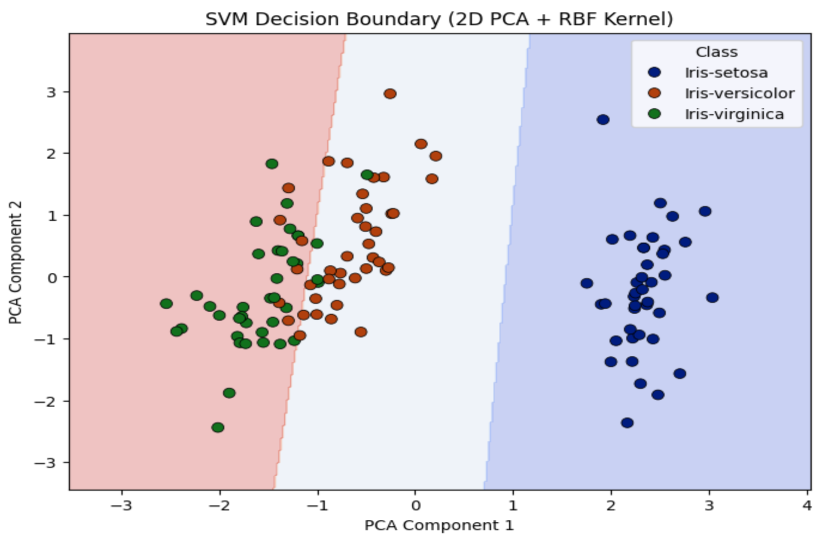
DATASET – https://archive.ics.uci.edu/dataset/53/iris

ALGORITHM USED – SUPPORT VECTOR MACHINE (SVC WITH RBF KERNEL)

WORKING:

1. Loaded the Iris dataset containing features like sepal length, sepal width, petal length, and petal width.
2. Extracted features (X) and target labels (y) from the dataset.
3. Applied log transformation to selected features to reduce skewness and improve model performance.
4. Scaled the features using StandardScaler to bring all features to the same scale.
5. Split the dataset into training and testing sets.
6. Initialized an SVM model with RBF kernel using SVC(kernel='rbf').
7. Trained the model on the scaled training data.
8. Made predictions on the test data.
9. Evaluated the model using accuracy score and confusion matrix.
10. SVM with RBF kernel was used because it handles non-linear classification by using Gaussian-based separation.
11. PCA FOR VISUALIZATION IN 2D

RESULTS:




```
Accuracy: 100.0
                 precision    recall  f1-score

    Iris-setosa       1.00      1.00      1.00
Iris-versicolor       1.00      1.00      1.00
 Iris-virginica       1.00      1.00      1.00

       accuracy                           1.00
```

TASK 7 - PCA on Digits Dataset

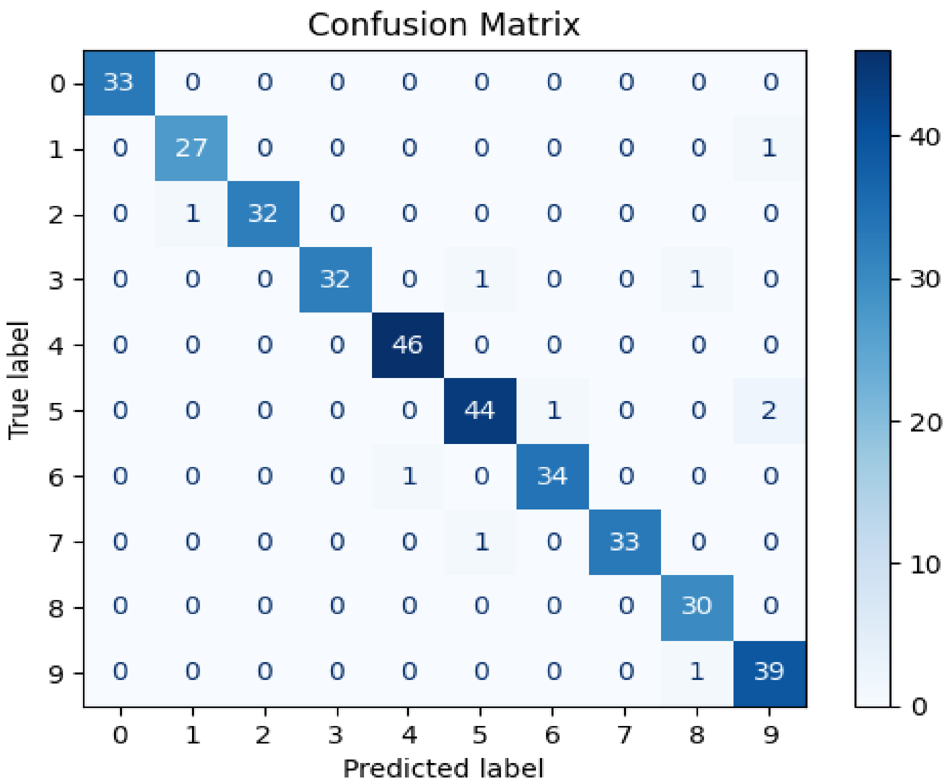
DATASET - https://scikit-learn.org/stable/auto_examples/classification/plot_digits_classification.html

AlGORITHM USED : LOGISTIC REGRESSION

WORKING :

1. Loaded the Digits dataset which contains 8×8 pixel images of handwritten digits (0 to 9) along with their labels.
2. Flattened each image from 8×8 into a 64-length vector to create the feature matrix
3. Applied log1p transformation (log(1 + x)) to the pixel values to reduce skewness and compress dynamic range.
4. Scaled the transformed data using StandardScaler to normalize the input features.
5. Split the dataset into training and testing sets for model evaluation.
6. Initialized a Logistic Regression model for multiclass classification.
7. Trained the model on the preprocessed training data.
8. Predicted digit classes on the test set using the trained model.
9. Evaluated model performance using accuracy score, classification report, and confusion matrix.
10. Logistic Regression was chosen for its simplicity and effectiveness in linear multiclass classification.

RESULTS :



ACCURACY : 97.22222222222222

TASK 8 - Bank Loan Default Prediction

DATASET – https://www.kaggle.com/datasets/wordsforthewise/lending-club

AIGORITHM USED : LOGISTIC REGRESSION

WORKING :

1. THE DATASET CONTAINS LOAN APPLICATION DETAILS AND THE TARGET VARIABLE INDICATES LOAN DEFAULT (CHARGEOFF OR FULLY PAID)
2. DROPPED UNIMPORTANT FEATURES HAVING MORE THAN 70% NULL VALUES TO AVOID BIAS AND REDUCE NOISE
3. APPLIED LABEL ENCODING TO CONVERT CATEGORICAL VARIABLES INTO NUMERICAL FORM
4. USED LIGHTGBM FEATURE IMPORTANCE TO IDENTIFY MOST RELEVANT FEATURES FOR MODELING
5. BALANCED THE DATASET TO HANDLE CLASS IMBALANCE BETWEEN DEFAULT AND NON-DEFAULT CLASSES
6. APPLIED LOGISTIC REGRESSION TO PREDICT THE PROBABILITY OF LOAN DEFAULT

$$P(Y=1|X) = \frac{1}{1+e^{-(B0+B1X1+B2X2+\ldots+BNXN)}}$$

RESULTS:

```
Validation Accuracy: 0.8278014359798747
              precision    recall  f1-score   support

           0       0.78      0.91      0.84    323923
           1       0.89      0.75      0.81    323022

    accuracy                           0.83    646945
   macro avg       0.84      0.83      0.83    646945
weighted avg       0.84      0.83      0.83    646945
```

TASK 9 - Prophet for Sales Forecasting

DATASET – https://www.kaggle.com/c/demand-forecasting-kernels-only/data

AIGORITHM USED : LIGHT GRADIENT BOOSTING MACHINE

WORKING :

1. **APPLIED THE LAG FEATURES TO TIME RELATED FEATURES**
2. **APPLIED EXPONENTIALLY WEIGHTED MOVING FEATURES BY CALLING LAG FUNCTION**
3. **FOR TRAIN TAKE ALL DATA BEFORE JAN 2017 AND FOR VALIDATION TAKE NEXT 3 MONTHS OF 2017**
4. **CALCULATE SMAPE (SYMMETRIC MEAN ABSOLUTE PERCENT ERROR) FOR LGBM**
5. **LIGHTGBM IS A GRADIENT BOOSTING FRAMEWORK THAT BUILDS AN ENSEMBLE OF DECISION TREES IN A SEQUENTIAL MANNER TO MINIMIZE LOSS. IT USES LEAF-WISE TREE GROWTH (UNLIKE XGBOOST'S LEVEL-WISE) FOR BETTER ACCURACY AND SPEED.**

RESULTS :

```
MAE: 0.1494
RMSE: 0.2033
R²: 0.8709
```

TASK 10 - Logistic vs Random Forest

DATASET - https://www.kaggle.com/c/titanic/data

AlGORITHM USED : LOGISTIC REGRESSION, RANDOM FOREST CLASSIFIER

WORKING :

1. Preprocessed the Titanic dataset by handling missing values and encoding categorical variables as in previous tasks.
2. Engineered features such as IsAlone, tot_family, and binned versions of Age and Fare.
3. Applied one-hot encoding to categorical features including Sex, Embarked, AgeBin, and FareBin.
4. Dropped irrelevant columns like Name, Ticket, and Cabin.
5. Trained both **Logistic Regression** and **Random Forest Classifier** on the same processed dataset.
6. Compared both models on evaluation metrics such as accuracy, precision, recall, and F1-score.
7. Compared performance with baseline models to check improvement using boosting.

RESULTS :

```
Logistic Regression Performance:
Accuracy: 0.9330143540669856
Precision: 0.8827160493827161
Recall: 0.9407894736842105
ROC-AUC: 0.9689107637514839
```

```
Random Forest Performance:
Accuracy: 0.7488038277511961
Precision: 0.6477987421383647
Recall: 0.6776315789473685
ROC-AUC: 0.8338073802928374
```

TASK 11 - Logistic vs Random Forest

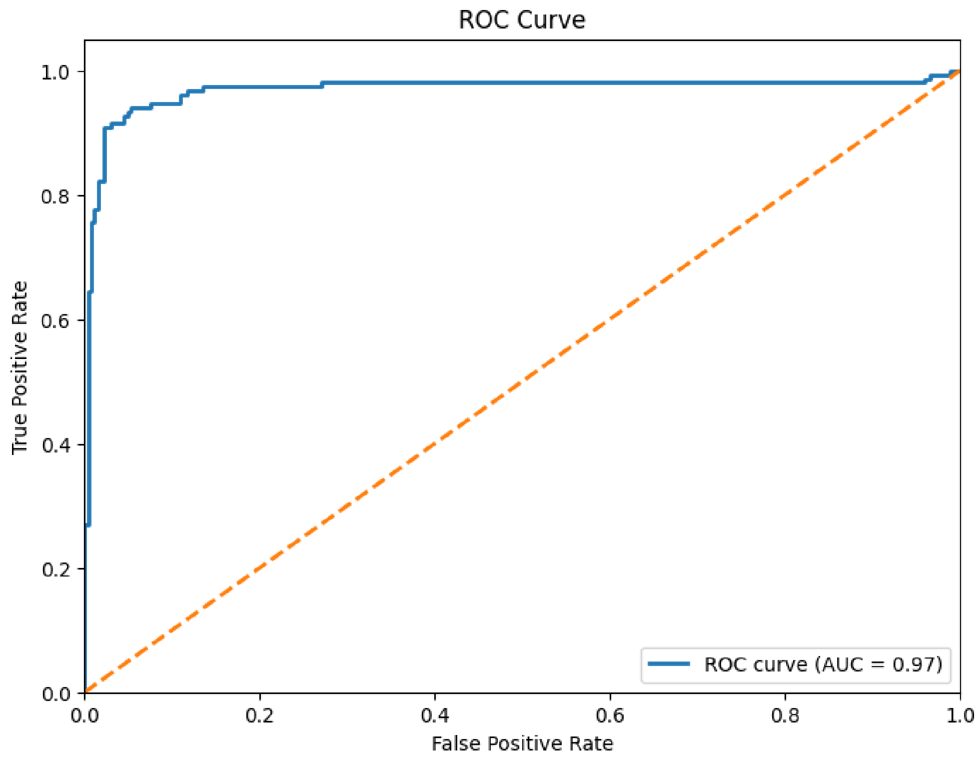
DATASET - https://www.kaggle.com/c/titanic/data

AIGORITHM USED : LOGISTIC REGRESSION

WORKING :

1. TRAINED CLASSIFICATION MODELS LOGISTIC REGRESSION ON TITANIC DATASET.
2. USED PREDICT_PROBA() OR DECISION_FUNCTION() TO GET MODEL PROBABILITIES.
3. COMPUTED TRUE POSITIVE RATE (TPR) AND FALSE POSITIVE RATE (FPR) AT VARIOUS THRESHOLDS USING ROC_CURVE().
4. PLOTTED THE ROC CURVE TO VISUALIZE THE MODEL'S ABILITY TO DISTINGUISH BETWEEN CLASSES.
5. CALCULATED THE AREA UNDER THE CURVE (AUC) TO MEASURE OVERALL CLASSIFICATION PERFORMANCE.
6. INTERPRETED THAT A MODEL WITH ROC-AUC CLOSE TO 1 HAS EXCELLENT SEPARABILITY, WHILE 0.5 INDICATES NO BETTER THAN RANDOM GUESSING.

RESULTS:



TASK 12 - GridSearchCV on Decision Tree

DATASET - https://archive.ics.uci.edu/dataset/53/iris

AlGORITHM USED : DECISION TREE

WORKING :

1. Loaded the Iris dataset containing features like sepal length, sepal width, petal length, and petal width.
2. COMPUTED TRUE POSITIVE RATE (TPR) AND FALSE POSITIVE RATE (FPR) AT VARIOUS THRESHOLDS USING ROC_CURVE().
3. NO TRANSFORMATION'S SCALED APPLIED BECAUSE IT IS TREE TYPE CLASSIFIER
4. INITIALIZED A DECISION TREE CLASSIFIER USING DECISIONTREECLASSIFIER().

5 DEFINED A PARAMETER GRID FOR TUNING (E.G., MAX_DEPTH, MIN_SAMPLES_SPLIT, CRITERION).

6. APPLIED GRIDSEARCHCV WITH CROSS-VALIDATION TO FIND THE BEST HYPERPARAMETERS.

7. TRAINED THE MODEL USING THE BEST PARAMETERS FROM GRID SEARCH.

8. MADE PREDICTIONS ON THE TEST DATA.

9. EVALUATED THE MODEL USING ACCURACY SCORE AND CONFUSION MATRIX.

10. DECISION TREE WITH GRID SEARCH WAS USED TO IMPROVE MODEL PERFORMANCE BY TUNING DEPTH AND SPLIT

RESULTS :

ACCURACY : 100 %

TASK 13 - Churn Prediction Model

DATASET - https://www.kaggle.com/datasets/blastchar/telco-customer-churn

AlGORITHM USED : LIGHT GBM

WORKING :

1. **CONVERTED THE NUMBERS TO NUMERIC COLUMNS**
2. **APPLIED SQRT TRANSFORMATION FOR PARTIALLY RIGHT SKEWED DATA**
3. **LABEL ENCODED THE CATEGORICAL DATA**
4. **SINCE IT IS TREE TYPE NO NEED OF SCALING**
5. **APPLIED THE LGBM'S LEAF TO TREE BOTTOM UP APPROACH**
6. **REUTURN THE PREDICTIONS**

RESULTS :

```
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.91      0.87      1035
           1       0.67      0.50      0.57       374

    accuracy                           0.80      1409
   macro avg       0.75      0.71      0.72      1409
weighted avg       0.79      0.80      0.79      1409
```

TASK 14 - DBSCAN on Synthetic Data

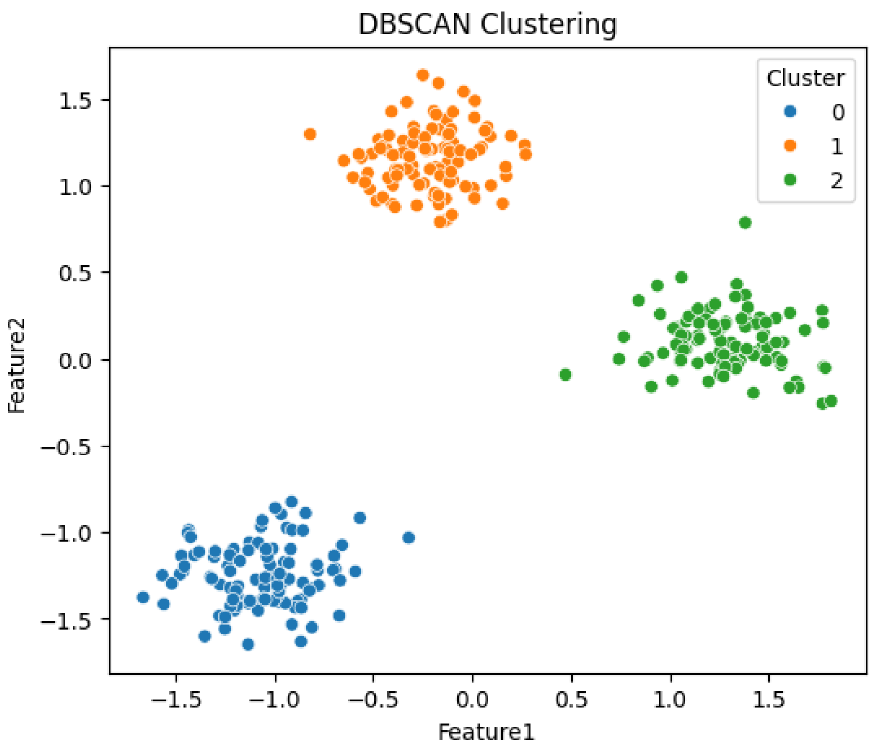
DATASET - https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html

AlGORITHM USED : Density-Based Spatial Clustering (KNN UNSUPERVISED)

WORKING :

1. A synthetic dataset was created using two numerical features for easy visualization.
2. Before clustering, the data was scaled using StandardScaler to ensure fair distance measurement.
3. DBSCAN was chosen for its ability to identify clusters of varying shape and to handle noise or outliers.
4. The model was initialized with suitable eps and min_samples values based on visual inspection or testing.
5. DBSCAN grouped the dense areas into clusters and marked low-density points as noise.
6. The results were visualized using a scatter plot where each cluster had a unique color, and outliers were highlighted.
7. DBSCAN worked well for this kind of irregular-shaped data compared to methods like KMeans.
8. This approach demonstrated how density-based clustering can reveal natural groupings in unlabeled data.

RESULTS:



Best DBSCAN Params: {'eps': 0.4, 'min_samples': 3}

Best Silhouette Score: 81.82412709664815

TASK 15 - KNN from Scratch

DATASET - https://archive.ics.uci.edu/dataset/53/iris

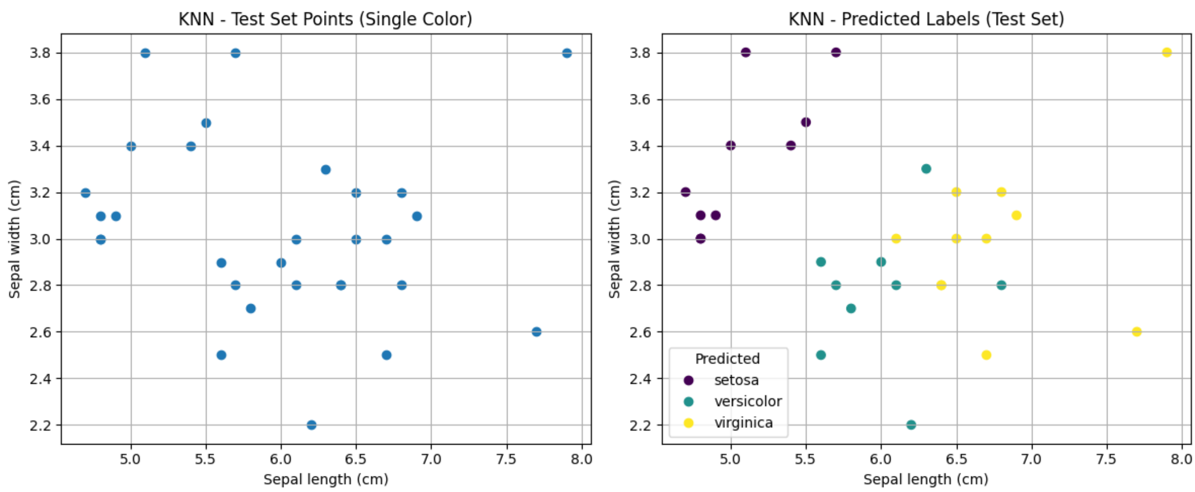
AlGORITHM USED : KNN (FROM SCRATCH)

WORKING :

1. The Iris dataset was loaded, containing features like sepal length, sepal width, petal length, and petal width, along with the target flower species.
2. Features (X) and labels (y) were extracted for use in the KNN algorithm.
3. Data was normalized using StandardScaler to ensure fair distance calculation.
4. The dataset was split into training and testing sets.
5. A KNN classifier was implemented manually without using sklearn's built-in KNeighborsClassifier.
6. The algorithm calculated the Euclidean distance from each test sample to all training samples.
7. For each test sample, the k closest neighbors were identified based on the smallest distances.
8. The class labels of those k neighbors were collected, and the most frequent label was selected as the predicted class.
9. The final predictions were compared with true labels to compute the accuracy of the model.
10. The model's performance was evaluated, and results showed how a simple distance-based algorithm can classify data effectively.

RESULTS:

ACCURACY :100



TASK 16 - Sentiment Analysis on Movie Reviews

DATASET - https://ai.stanford.edu/~amaas/data/sentiment/

AlGORITHM USED : LOGISTIC REGRESSION + TFIDF

WORKING :

1. The dataset contains thousands of movie reviews labeled as positive or negative.
2. Each review text was first cleaned by removing unwanted HTML tags using BeautifulSoup.
3. The cleaned text was converted to lowercase to maintain consistency.
4. All non-alphabetic characters were removed using regular expressions to reduce noise.
5. The text was then tokenized into individual words using NLTK's word tokenizer.
6. Stopwords were removed to retain only meaningful words.
7. Lemmatization was applied to reduce each word to its base form for better generalization.
8. Only tokens longer than one character were kept to remove unnecessary short words.
9. Finally, the cleaned tokens were joined back into a string to form the final preprocessed review.
10. These cleaned reviews were then used to train a machine learning classifier Logistic for sentiment prediction.

RESULTS:

```
Accuracy: 0.88288
              precision    recall  f1-score   support

    Negative       0.89      0.88      0.88     12500
    Positive       0.88      0.89      0.88     12500

    accuracy                           0.88     25000
   macro avg       0.88      0.88      0.88     25000
weighted avg       0.88      0.88      0.88     25000
```

TASK 17 - CNN Image Classifier

DATASET - https://www.cs.toronto.edu/~kriz/cifar.html

AlGORITHM USED : CNN

WORKING :

1. The CIFAR-10 dataset contains 60,000 color images of size 32x32 across 10 different classes.
2. The data was loaded and split into training and testing sets.
3. Pixel values were normalized to bring them into a standard range of 0 to 1.
4. A CNN model was built using convolutional layers, ReLU activation, and max pooling layers.
5. Flatten and dense layers were added to convert features into class predictions.
6. Softmax activation was used in the output layer to handle multiclass classification.
7. The model was compiled with categorical crossentropy loss and Adam optimizer.
8. The CNN was trained for multiple epochs on the training data.
9. Accuracy and loss were monitored using the validation set.

   CNN was chosen because it is highly effective for extracting spatial features from image data.

RESULTS:

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 32)	896
batch_normalization (BatchNormalization)	(None, 32, 32, 32)	128
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_1 (Conv2D)	(None, 16, 16, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 16, 16, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 64)	0
conv2d_2 (Conv2D)	(None, 8, 8, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 8, 8, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 128)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 128)	262,272
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1,290

	precision	recall	f1-score
0	0.77	0.81	0.79
1	0.83	0.90	0.86
2	0.75	0.63	0.68
3	0.65	0.52	0.58
4	0.77	0.71	0.74
5	0.74	0.66	0.70
6	0.72	0.89	0.80
7	0.78	0.84	0.81
8	0.87	0.86	0.86
9	0.81	0.88	0.84
cy			0.77

TASK 18 - Credit Card Fraud Detection

DATASET - https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

AlGORITHM USED : RNN

WORKING :

1. The dataset contains credit card transactions with 284,807 entries and 31 features, including anonymized PCA components and the transaction amount.
2. The target variable indicates whether the transaction is fraudulent (1) or not (0).
3. Class distribution was highly imbalanced, with fraud cases being very rare.
4. The data was scaled to bring features into a uniform range using StandardScaler.
5. Each transaction was reshaped to match the RNN input format (samples, timesteps, features).
6. An RNN model was built using LSTM or SimpleRNN layers to learn temporal patterns in the data.
7. Dense layers were added after the RNN layer for binary classification.
8. The model was compiled using binary crossentropy loss and Adam optimizer.
9. Training was done on the processed data while monitoring validation accuracy.

Layer (type)	Output Shape	Param #
simple_rnn_2 (SimpleRNN)	(None, 1, 64)	6,080
dropout_2 (Dropout)	(None, 1, 64)	0
simple_rnn_3 (SimpleRNN)	(None, 32)	3,104
dropout_3 (Dropout)	(None, 32)	0
dense_1 (Dense)	(None, 1)	33

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
...
    accuracy                           1.00     56962
   macro avg       0.92      0.89      0.91     56962
weighted avg       1.00      1.00      1.00     56962
```

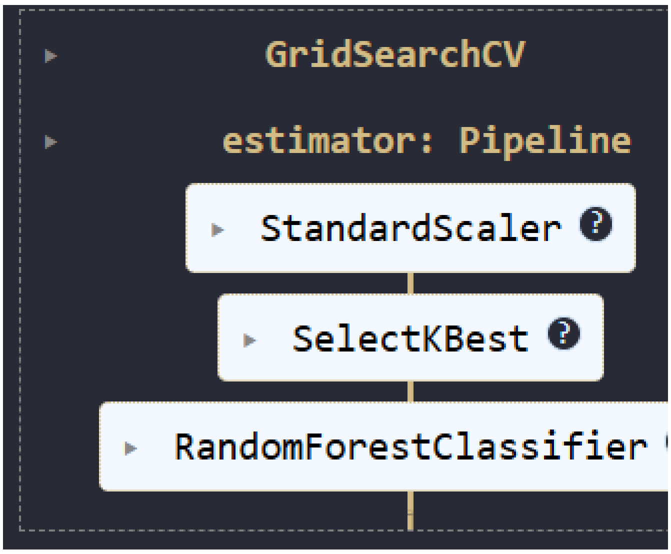
TASK 19 - Classification Pipeline

DATASET - https://archive.ics.uci.edu/dataset/53/iris

AIGORITHM USED : RANDOM FOREST

WORKING :

1. A machine learning pipeline was created to streamline preprocessing, feature selection, and model training.
2. The data was first scaled using StandardScaler to normalize input features for consistent performance.
3. SelectKBest was used for feature selection to retain the top K most relevant features based on statistical tests.
4. A RandomForestClassifier was applied as the main classification model to handle non-linearity and feature importance.
5. The entire pipeline was wrapped inside GridSearchCV to perform hyperparameter tuning and find the best combination of parameters.
6. Grid search tested different values for K (number of features), and Random Forest parameters like n_estimators and max_depth.
7. Cross-validation was used internally to ensure generalization and reduce overfitting.
8. The best model and parameter set were selected based on validation accuracy or scoring metric.
9. This method ensures a clean workflow where preprocessing and model tuning happen in a single integrated pipeline.




```
Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      0.90      0.95        10
   virginica       0.91      1.00      0.95        10

    accuracy                           0.97        30
```

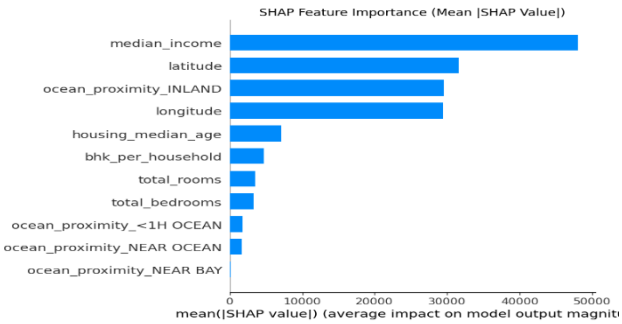
TASK 20 - SHAP Explainability on Random Forest

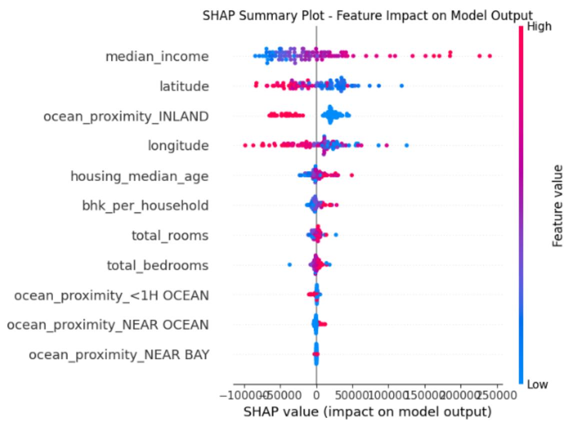
DATASET - https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data

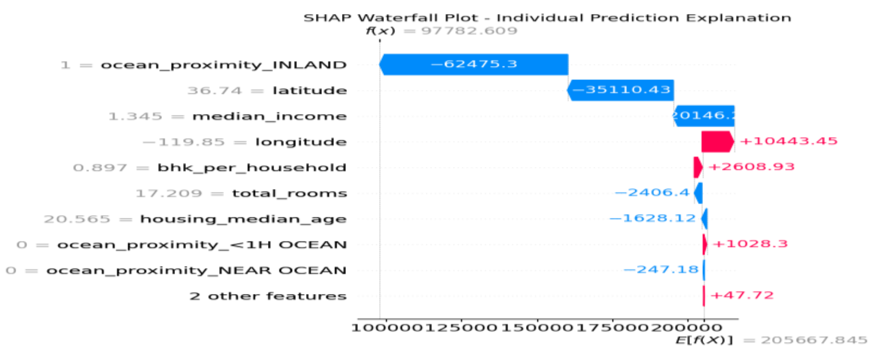
AlGORITHM USED : RANDOM FOREST

WORKING :

1. **A Random Forest Regressor was trained on the processed dataset to predict house prices.**
2. **SHAP (SHapley Additive exPlanations) was used to explain individual predictions and overall feature importance.**
3. **SHAP values represent the contribution of each feature to the model's prediction, based on cooperative game theory.**
4. **The Waterfall Plot was used to show how individual feature values pushed a prediction higher or lower from the average prediction.**
5. **The Summary Plot combined feature importance and value effect — showing both impact and direction for every feature.**
6. **High SHAP values in red showed features pushing the prediction up, and low values in blue pushed it down.**
7. **The Bar Plot of mean SHAP values ranked features by their average impact on model output across all predictions.**
8. **Key features like median_income, latitude, ocean_proximity, and longitude had the highest influence on prediction.**
9. **SHAP provided clear insight into how the Random Forest made its decisions, making the model transparent and trustworthy.**
10. **This explainability technique is especially useful for stakeholders who need to understand why a model predicts a certain outcome.**







TASK 21 - AutoML with TPOT

DATASET - https://www.kaggle.com/c/titanic/data

AlGORITHM USED : MULTI (TPOT)

WORKING :

1. The Titanic dataset was used to predict survival outcomes based on features like age, sex, class, and fare.
2. TPOT (Tree-based Pipeline Optimization Tool) was used to automate the entire machine learning pipeline process.
3. TPOT uses genetic programming to search through different model and preprocessing combinations.
4. The TPOTClassifier was initialized with 10 generations and a population size of 50.
5. This means TPOT tested 50 different model pipelines per generation, evolving the best ones for 10 rounds.
6. The model was trained on the Titanic dataset with features as input and survival status as the target.
7. TPOT internally tried various models, scalers, selectors, and parameters.
8. In this case, TPOT selected a Decision Tree as the best performing model pipeline.
9. The entire search was run in parallel using all CPU cores (n_jobs=-1) to speed up processing.
10. TPOT helped save manual effort by automatically choosing the best model and preprocessing steps.
11. This makes TPOT useful for quick prototyping, benchmarking, and exploring optimized pipelines without manual tuning.

```
Generation 1 - Current best internal CV score: 0.8372920720607621

Generation 2 - Current best internal CV score: 0.8372920720607621

Generation 3 - Current best internal CV score: 0.8372920720607621

Generation 4 - Current best internal CV score: 0.8372920720607621

Generation 5 - Current best internal CV score: 0.8383717280773334

Generation 6 - Current best internal CV score: 0.838390559286925

Generation 7 - Current best internal CV score: 0.838390559286925

Generation 8 - Current best internal CV score: 0.838390559286925

Generation 9 - Current best internal CV score: 0.838390559286925

Generation 10 - Current best internal CV score: 0.8395141547925429

Best pipeline: RandomForestClassifier(PolynomialFeatures(input_matrix, degree=2,
```

```
TPOTClassifier

TPOTClassifier(generations=10, n_jobs=-1, population_size=50, random_state=42,
               verbosity=2)
```

TASK 22 - GridSearchCV on Decision Tree

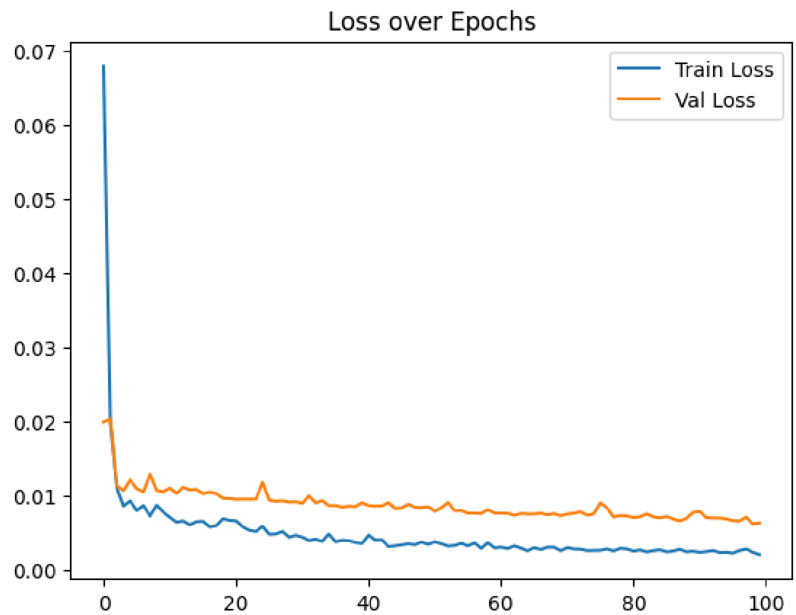
DATASET - https://finance.yahoo.com/

AlGORITHM USED : LSTM

WORKING :

1. Historical stock price data was downloaded from Yahoo Finance.
2. Only the ‘Close’ prices were selected for forecasting future stock trends.
3. The data was scaled using MinMaxScaler to bring values into the 0–1 range.
4. A sliding window approach was used to create sequences of past prices as input and the next value as the target.
5. The input data was reshaped into 3D format as required by LSTM (samples, time steps, features).
6. An LSTM model was built with layers such as LSTM, Dropout, and Dense for sequential learning.
7. The model was compiled with mean squared error (MSE) loss and Adam optimizer.
8. Training was done on historical data, and predictions were made on test data.
9. The predicted values were inverse transformed to get actual price scale.
10. Model performance was evaluated using RMSE or MAE.
11. LSTM was chosen because it captures long-term dependencies and trends in time series data.

```
 Next 10 days predictions:
Day 1: $205.34
Day 2: $206.46
Day 3: $207.08
Day 4: $207.44
Day 5: $207.69
Day 6: $207.87
Day 7: $208.02
Day 8: $208.15
Day 9: $208.28
Day 10: $208.40
```



TASK 23 - Transformers with HuggingFace

DATASET - https://ai.stanford.edu/~amaas/data/sentiment/

AlGORITHM USED : BERT

WORKING :

1. The IMDB movie review dataset was used for binary sentiment classification (positive or negative).
2. Text data was preprocessed by removing HTML tags, special characters, and unnecessary whitespace.
3. The dataset was tokenized using a pretrained BERT tokenizer from the Hugging Face Transformers library.
4. Each review was converted into input IDs and attention masks compatible with the BERT model.
5. A pretrained BERT base model (bert-base-uncased) was loaded and fine-tuned for sequence classification.
6. The model was trained using a small learning rate with AdamW optimizer and cross-entropy loss.
7. Training included batch processing with attention to GPU memory usage and sequence length.
8. The model output was passed through a dense layer with softmax activation to predict sentiment classes.
9. Predictions were evaluated using accuracy, precision, recall, and F1-score.
10. Transformers like BERT are powerful because they capture context and meaning from the full sentence using self-attention.
11. This approach outperforms traditional NLP models on benchmark tasks like IMDB sentiment analysis.

Final Test Accuracy: 0.8799

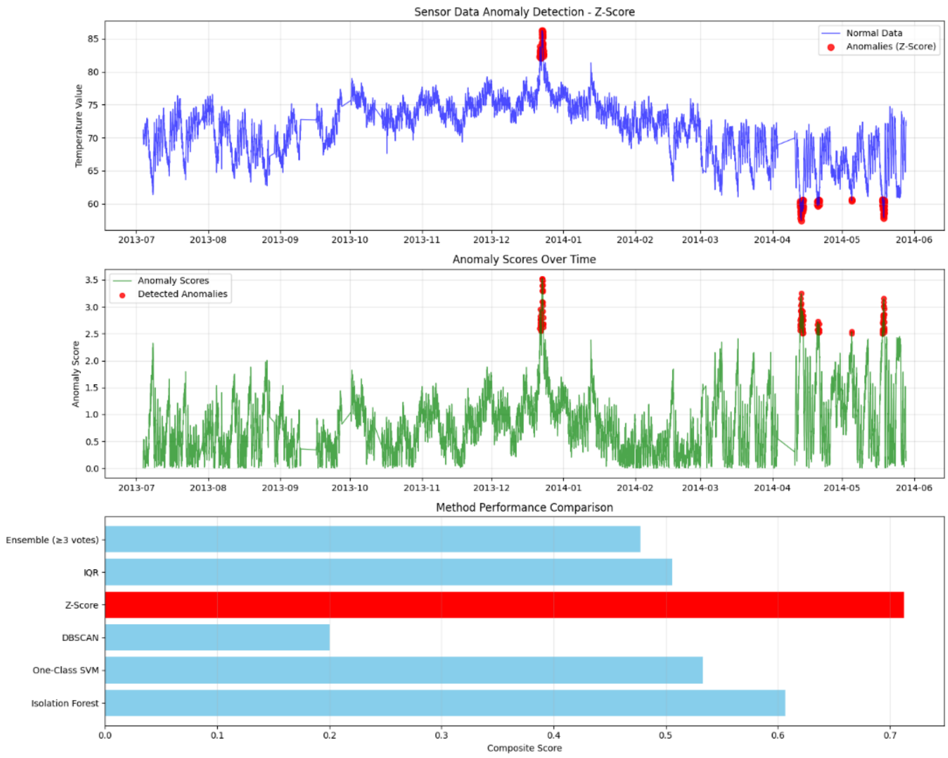
TASK 24 - Sensor Data Anomaly Detection

DATASET - https://github.com/numenta/NAB/blob/master/data/realKnownCause/ambient_temperature_system_failure.csv

AlGORITHM USED : ISOLATION FOREST , SVM , DBSCAN

WORKING :

1. Sensor data containing ambient temperature over time was analyzed to detect system failures and anomalies.
2. The Z-Score method was used to detect statistical outliers by identifying values beyond a threshold standard deviation.
3. Isolation Forest identified anomalies by isolating rare patterns with fewer random splits in the tree structure.
4. One-Class SVM attempted to learn the boundary of normal data and detected deviations as anomalies.
5. DBSCAN attempted density-based detection but failed due to sparsity in the time series.
6. The Interquartile Range (IQR) method marked outliers that fall outside the range of Q1−1.5×IQR and Q3+1.5×IQR.
7. An ensemble method was used to combine results from all detectors, flagging anomalies confirmed by at least 3 methods.
8. Visualizations showed anomalies marked on top of raw sensor data and over time using anomaly scores.
9. The method comparison plot and summary table showed performance across metrics like precision, recall, F1-score, and coverage.
10. Z-Score showed the highest precision and recall for this dataset, while DBSCAN over-predicted all points as anomalies.
11. The ensemble method balanced between false positives and missed detections, making it more robust in real scenarios.



TASK 25 - Intent Classification for Chatbots

DATASET - https://github.com/clinc/oos-eval

AlGORITHM USED : RANDOM FOREST , TFIDF

WORKING :

1. The Clinc OOS evaluation dataset contains user utterances labeled as either in-domain (IN) or out-of-scope (OOS).
2. TF-IDF vectorization was used to convert raw text into numerical feature vectors based on term frequency and inverse document frequency.
3. These vectors captured the importance of words in each utterance relative to the dataset.
4. A Random Forest classifier was trained on the TF-IDF features to learn patterns of known (in-scope) intents.
5. During evaluation, if the classifier was confident about its prediction, the input was considered in-scope.
6. If confidence was low or the predicted intent didn't match expected labels, it was marked as OOS (out-of-scope).
7. The goal was to build a system that can not only classify known intents but also reject unknown or unsupported queries.
8. Model performance was evaluated using accuracy, F1-score (IN), and OOS detection rate.

```
Text: 'Can you help me with my account?'
  Prediction: in-scope (confidence: 0.557)
  Probabilities: Out-of-scope=0.443, In-scope=0.557
Text: 'What's the meaning of life?'
  Prediction: in-scope (confidence: 0.521)
  Probabilities: Out-of-scope=0.479, In-scope=0.521
Text: 'I need to cancel my subscription'
  Prediction: in-scope (confidence: 0.542)
  Probabilities: Out-of-scope=0.458, In-scope=0.542
Text: 'Sing me a song in Japanese'
  Prediction: in-scope (confidence: 0.503)
  Probabilities: Out-of-scope=0.497, In-scope=0.503
Text: 'How do I reset my password?'
  Prediction: out-of-scope (confidence: 0.505)
  Probabilities: Out-of-scope=0.505, In-scope=0.495
```

	precision	recall	f1-score	support
Out-of-scope (0)	0.13	0.82	0.22	240
In-scope (1)	0.99	0.70	0.82	4500
accuracy			0.71	4740

TASK 26 - XGBoost Multi-Class Classification

DATASET - https://archive.ics.uci.edu/dataset/53/iris

AlGORITHM USED : XGBOOST

WORKING :

1. The Iris dataset contains 150 flower samples with 4 features: sepal length, sepal width, petal length, and petal width.
2. The goal was to classify each flower into one of three species: Setosa, Versicolor, or Virginica.
3. Data was split into input features (X) and target labels (y).
4. The target labels were encoded into numeric format suitable for classification.
5. An XGBoost classifier was initialized with objective='multi:softmax' to handle multi-class output.
6. The model was trained using the training data and validated on the test set.
7. XGBoost used boosting over decision trees to optimize for accuracy by minimizing classification error.
8. The final predictions were compared to the true labels using accuracy score and confusion matrix.
9. XGBoost was chosen for its speed, regularization capabilities, and robustness on structured data.

```
Classification Report:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       0.90      0.90      0.90        10
 Iris-virginica       0.90      0.90      0.90        10

       accuracy                           0.93        30
      macro avg       0.93      0.93      0.93        30
   weighted avg       0.93      0.93      0.93        30
```

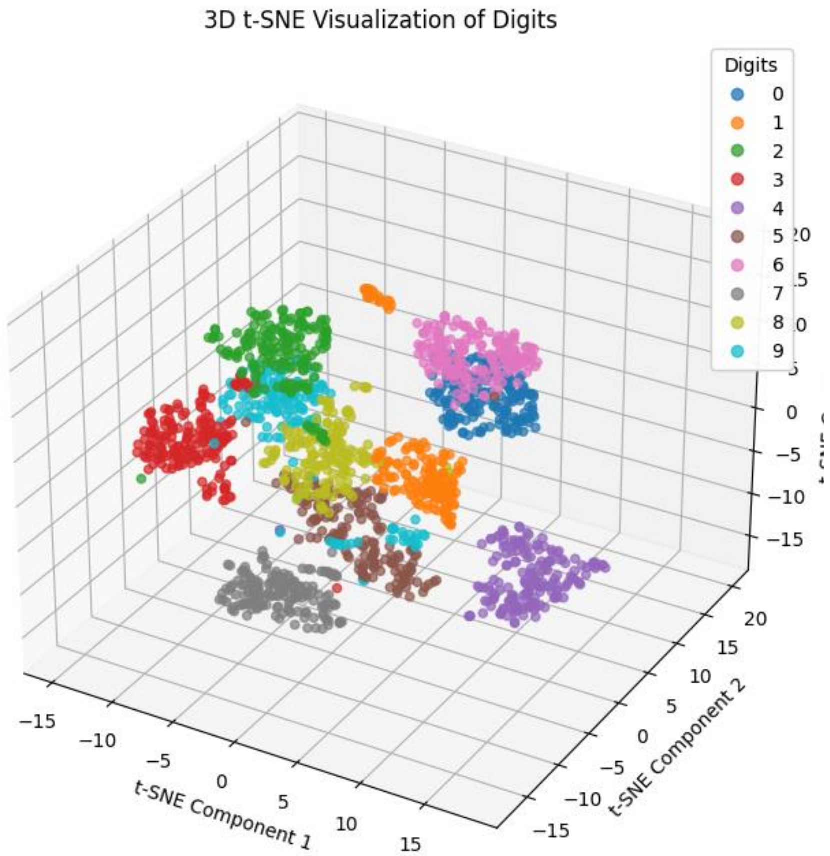
TASK 27 - Visualize Data using t-SNE

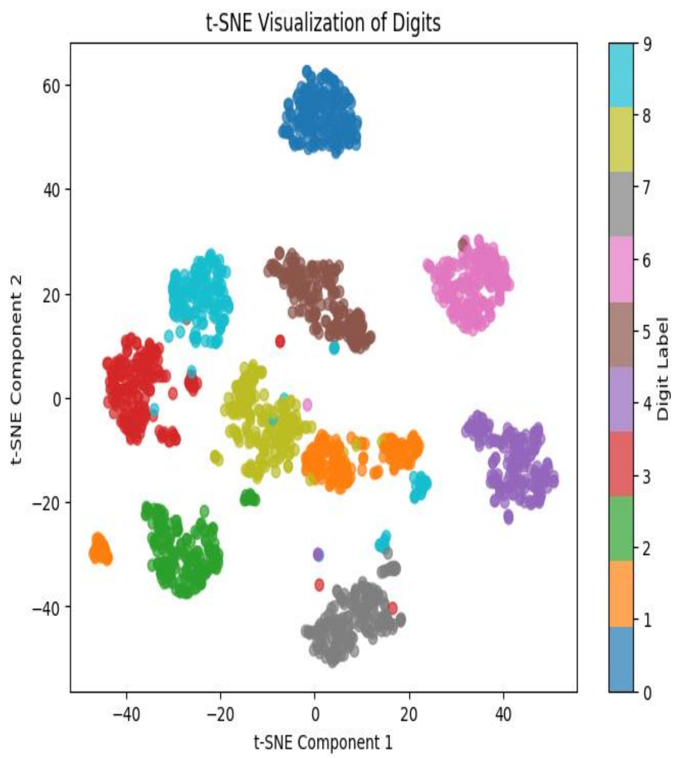
DATASET - https://scikit-learn.org/stable/auto_examples/classification/plot_digits_classification.html

AlGORITHM USED : T-SNE

WORKING :

1. Used the Digits dataset consisting of 1797 images of handwritten digits (0 to 9).
2. Each image is an 8×8 grayscale matrix, flattened into a 64-dimensional feature vector.
3. No preprocessing or scaling was applied before running t-SNE.
4. Applied 2D t-SNE to reduce the 64D data to 2D while preserving neighborhood structure.
5. Plotted the transformed data with each point color-coded by its digit label.
6. Observed clear separation of clusters corresponding to different digits.
7. Also applied 3D t-SNE and visualized the embeddings using a 3D scatter plot.
8. Clusters in 3D also showed distinct groupings of digits, useful for visual inspection of separability.
9. t-SNE was chosen as it effectively handles non-linear dimensionality and provides intuitive visual groupings.
10. This technique helps understand underlying data structure and detect potential class overlaps.





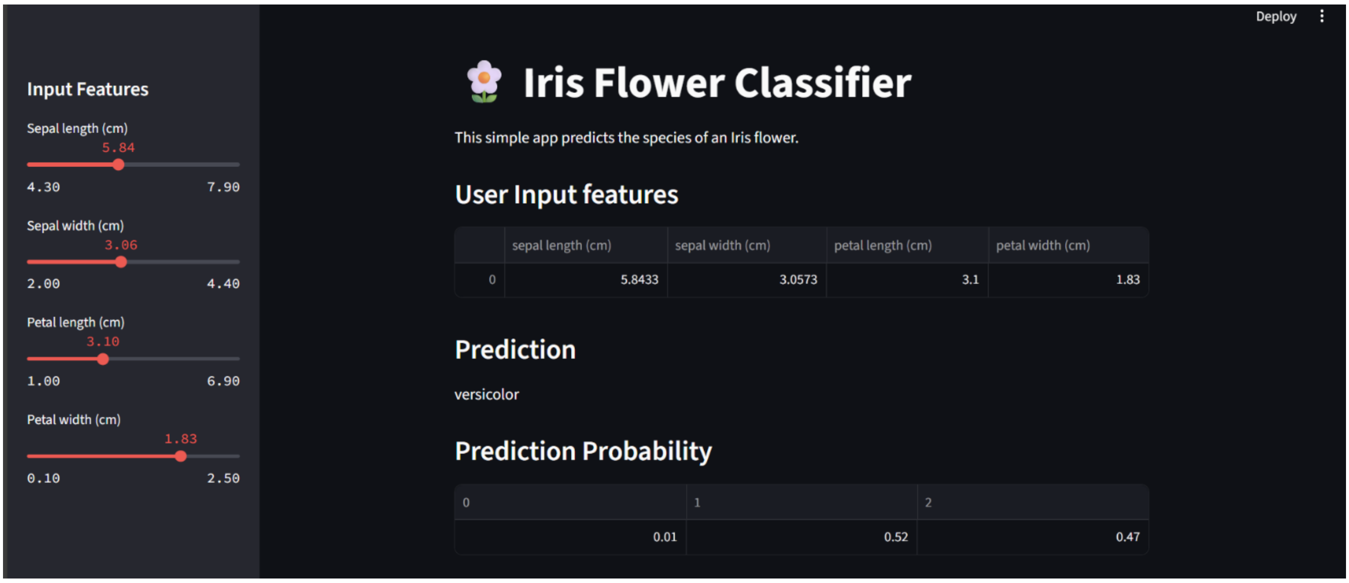
TASK 28 - Streamlit App Deployment

DATASET - https://archive.ics.uci.edu/dataset/53/iris

AlGORITHM USED : STREAMLIT WITH RANDOM FOREST

WORKING :

1. The Iris dataset was used, containing features like sepal length, sepal width, petal length, and petal width.
2. A Random Forest classifier was trained to predict the species of a flower based on input features.
3. The trained model was saved using joblib or pickle.
4. A Streamlit app was built to allow users to input feature values through sliders or number inputs.
5. The app loaded the trained Random Forest model and used it to make real-time predictions.
6. Upon clicking "Predict", the model output was displayed, showing the predicted flower species.
7. Additional elements like data visualizations, probability scores, and confusion matrix were optionally added.
8. The interface was designed to be simple, responsive, and usable without coding knowledge.
9. Streamlit enabled rapid deployment of the machine learning model as a lightweight web app.
10. The app was deployed either locally or to the cloud (e.g., Streamlit Cloud or Heroku).
11. This task demonstrated the integration of machine learning with user-facing web interfaces using Streamlit.



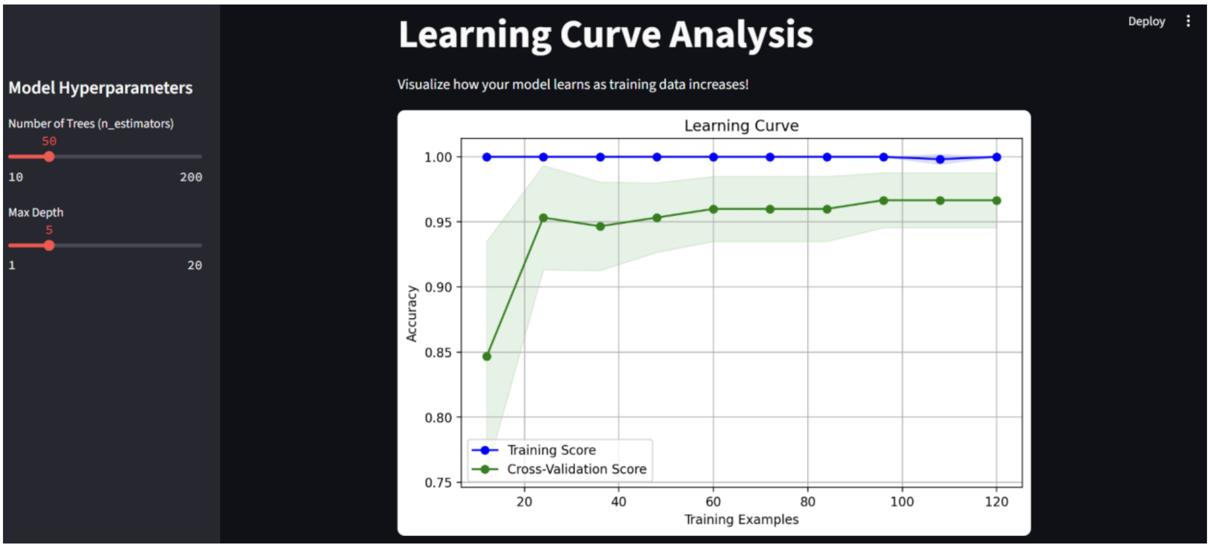
TASK 29 - Learning Curve Analysis

DATASET - https://archive.ics.uci.edu/dataset/53/iris

AIGORITHM USED : STREAMLIT, RANDOM FOREST

WORKING :

1. The Iris dataset was used for multi-class classification with features like sepal and petal dimensions.
2. A Random Forest model was trained with increasing amounts of training data to observe how performance changes.
3. Used learning_curve from scikit-learn to compute training and validation scores across various training sizes.
4. Plotted these scores to visualize the learning curve, showing how the model generalizes.
5. The curve helps identify underfitting (both low scores) or overfitting (large gap between train/test).
6. The learning curve was displayed inside a Streamlit app for real-time interaction and interpretation.
7. Streamlit allowed users to control parameters such as number of estimators, training size, and scoring metric.
8. The app updated the learning curve plot dynamically based on user inputs.
9. This interactive analysis helped in diagnosing model performance issues early and adjusting accordingly.
10. It is especially useful in determining if collecting more data would improve model performance.



TASK 30 - Ensemble Voting Classifier

DATASET - https://www.kaggle.com/c/titanic/data

AlGORITHM USED : ENSEMBLE VOTING (SOFT + HARD)

WORKING :

1. The Titanic dataset was used for binary classification – predicting survival based on passenger details.
2. Features like Pclass, Sex, Age, Fare, and Embarked were selected after handling missing values and encoding categories.
3. Three different models were used as base learners:
   – Logistic Regression
   – Random Forest
   – K-Nearest Neighbors
4. A VotingClassifier was created using these models to combine their predictions.
5. Hard voting was used to take the majority class prediction from all three models.
6. Soft voting was used to average predicted probabilities and select the most probable class.
7. Both versions were trained on the same dataset and evaluated on a test split using accuracy score.
8. Soft voting generally performed better due to its use of class probabilities, especially with well-calibrated models.
9. This ensemble approach improved prediction stability and reduced individual model biases.
10. Voting ensembles are simple, yet powerful, and easy to implement using sklearn.ensemble

Hard Voting Accuracy: 84.92822966507177

Soft Voting Accuracy: 84.92822966507177