***PROJECT 2:***

***LINEAR REGRESSION:***

***From sklearn.linear\_model import LinearRegression***

Model – Linearregression()

Model.fit(x\_train, y\_train)

Ypred =model.predict(x\_test)

Data = pd.DataFrame(data=(“Predicted Profit”:ypred.flatten()])

Training\_data\_model\_score = model.score(x\_train, y\_train)

Print(f”model Score/Performance on Training data (training\_data\_model\_score:.3f1”)

Testing data model score model.score(x\_ test, y\_test)

Print(f”model Score/Performance on Testing data (testing data model score: .3f)”)

✅️O.Os

Model Score/Performance on Training data 0.932

Model Score/Performance on Testing data 0.982

R2Score= r2\_score(ypred, y\_test)

Mse-mean\_squared\_error(ypred, y\_test)

Rmse=np.sqrt(mean\_squared\_error (ypred, y\_test))

Print(“Mean Squared Error is:”, mse)

Print(“Root Mean Squared Error is:”, rmse)

Print(“R-Squared score of model is : “, r2Score)

✅️0.Os،

Mean Squared Error is: 30114660.86441897

Root Mean Squared Error is: 5487.682649754718

R-Squared score of model is : 0.9822165488084384

***GRADIENT BOOSTING REGRESSION:***

Gradient\_boosting – GradientBoostingRegressor(n\_estimators=100, learning\_rate-0.1, random\_state-42)

Gradient\_boosting.fit(x\_train, y\_train)

# Making predictions on the test set

Y\_pred\_gb – gradient\_boosting.predict(x\_test)

# Calculating evaluation metrics on the test set

Mse\_gb = mean\_squared\_error(y\_test, y\_pred\_gb)

R2\_gb = r2\_score(y\_test, y\_pred\_gb)

Print(f”Mean Squared Error : (mse\_gb:.2f)”)

Print(f”R-squared : /r2\_gb:.2f)”)

✅️0.Os

Mean Squared Error : 83392317.94

R-squared : 0.95

Plt.figure(figsize-(8, 6))

Plt.scatter(y\_test, y\_pred\_svr, color-‘blue’, label-‘Actual Data’)

Plt.plot(y\_test, y\_test, color=’red”, linestyle=’-‘, linewidth=2, label=’Predicted Data’)

Plt.title(‘Support Vector Regression”)

Plt.legend()

Plt.grid(True)

Plt.show()

✅️0.1s

***RANDOM FOREST REGRESSION:***

# Random Forest

Random\_forest\_regressor = RandomForestRegressor(n\_estimators=100, random\_state=42)

Random\_forest\_regressor.fit(x\_train, y\_train)

Y\_pred\_rf = random\_forest\_regressor.predict(x\_test)

Mse\_rf – mean\_squared\_error(y\_test, y\_pred\_rf)

R2\_rf = r2\_score(y\_test, y\_pred\_rf)

Print(“Mean Squared Error: “ , mse\_rf)

Print(“R-sqaured: “, r2\_rf)

✅️ 0.1s

Mean Squared Error: 64932980.87782086

R-sqaured: 0.9621570742392656

Plt.figure(figsize=(8, 6))

Plt.scatter(y\_test, y\_pred\_dt, color-‘blue’, label-‘Actual Data’)

Plt.plot(y\_test, y\_test, color=’red’, linestyle=’-“, linewidth=2, label=’Predicted Data’)

Plt.title(‘Decision Tree Regression’)

Plt.legend()

Plt.grid(True)

Plt.show()

✅️0.1s

***LOGISTIC REGRESSION:***

X\_train, X\_test, y\_train, y\_test – train\_test\_split(x, y, test\_size=0.2, random\_state=42)

✅️0.Os

Logistic\_model – LogisticRegression(max\_iter-100)

# Train

Logistic\_model.fit(X\_train, y\_train)

# Predict

Y\_pred – logistic\_model.predict(X\_test)

# Evaluation

Accuracy = accuracy\_score(y\_test, y\_pred)

Conf\_matrix = confusion\_matrix(y\_test, y\_pred)

Classification\_rep = classification report(y\_test, y\_pred)

Print(f”Accuracy: (accuracy)”)

Print(f”Confusion Matrix:\n/conf\_matrix)”)

Print(f”Classification Report:\nfclassification\_rep)”)

✅️0.0s

Accuracy: 0.7467532467532467

Confusion Matrix:

[[78 21]

[18 37]]

**PROJECT2:**

Classification Report:

Precision

Recall f1-score

Support

1

0.81

0.64

0.79

0.67

0.80

0.65

9g

55

Accuracy

Macro avg

Weighted avg

0.75

0.73

0.75

154

154

154

0.73

0.75

0.73

0.75

***DECISION TREE CLASSIFIER:***

Tree\_classifier = DecisionTreeClassifier (max\_depth=5, random\_state=42)

# Train

Tree\_classifier.fit(X\_train, y\_train)

# Predict

Y\_pred – tree\_classifier.predict(X\_test)

# Evaluate

Accuracy = accuracy\_score(y\_test, y\_pred)

Conf matrix – confusion matrix(y\_test, y\_pred)

Classification\_rep – classification\_report(y\_test, y\_pred)

Print(f”Accuracy: (accuracy)”)

Print(f”Confusion Matrix:\n(conf\_matrix)”)

Print(f”Classification Report:\n(classification\_rep)”)

✅️0.Os

Accuracy: 0.7922077922077922

Confusion Matrix:

[[87 12]

[20 35]]

Classification Report:

Precision

Recall f1-score

Support

O

1

0.81

0.74

0.88

0.64

0.84

0.69

99

55

Accuracy

Macro avg

Weighted avg

0.79

0.77

0.79

154

154

154

0.78

0.79

9.76

0.79

***RANDOM FOREST CLASSIFIER:***

Rf\_classifier = RandomForestClassifier(n\_estimators-100, random\_state=42)

# Train

Rf\_classifier.fit(X\_train, y\_train)

# Predict

Y\_pred = rf\_classifier.predict(X\_test)

# Evaluate

Accuracy\_score(y\_test, y\_pred)

Accuracy

Conf\_matrix = confusion\_matrix(y\_test, y\_pred)

Classification \_rep = classification report(y\_test, y\_pred)

Print(f”Accuracy: (accuracy)”)

Print(f”Confusion Matrix:\n(conf\_matrix)”)

Print(f”Classification Report:\n(classification\_rep!”)

✅️0.2s

Accuracy: 0.7207792207792207

Confusion Matrix:

[[77 22]

[21 34]]

Classification Report:

Precision

Recall f1-score

Support

0 1

0.79

0.61

0.78

0.62

0.78

0.61

99

55

Accuracy

Macro avg

Weighted avg

0.72

0.70

0.72

154

154

154

0.70

0.72

0.70

0.72

***NAÏVE BAYES CLASSIFIER:***

Nb classifier = GaussianNB()

# Train

Nb\_classifier.fit(X\_train, y\_train)

# Predict

Y\_pred = nb\_classifier.predict(X\_test)

# Evaluate

Accuracy accuracy\_score(y\_test, y\_pred)

Conf\_matrix – confusion\_matrix(y\_test, y\_pred)

Classification\_rep = classification\_report(y\_test, y\_pred)

Print(f”Accuracy: (accuracy)”)

Print(f”Confusion Matrix:\n/conf\_matrix)”)

Print(f”Classification Report:\n(classification\_rep)”)

✅️0.0s

Accuracy: 0.7662337662337663

Confusion Matrix:

[[79 20]

[16 39]]

Classification Report:

Precision

Recall f1-score

Support

0

1

0.83

0.66

0.80

0.71

0.81

0.68

99

55

Accuracy

Macro avg

Weighted avg

0.77

0.75

0.77

154

154

154

0.75

0.77

0.75

0.77

***GRADIENT BOOSTING (XGBOOST):***

# Create DMatrix for XGBoost

Dtrain – xgb.DMatrix(X\_train, label-y\_train)

Dtest – xgb.DMatrix(X\_test, label-y\_test)

# Parameters for the XGBoost model

Params=f

Max\_depth’: 3, # Depth of trees

‘learning. Rate’: 0.1,

# Binary classification

“objective’: binary:logistic

Eval\_metric’: ‘logloss’ # Evaluation metric to use

# Train

Num\_round

Xgb\_model

100 # Number of boosting rounds

Xgb.train(params, dtrain, num\_round)

# Predict

Y\_pred – xgb\_model.predict(dtest)

Y\_pred\_binary – [1 if x >= 0.5 else 0 for x in y\_pred] # Converting probabilities to binary predictions

# Evaluate

Accuracy – accuracy\_score(y\_test, y\_pred\_binary)

Conf\_matrix = confusion\_matrix(y\_test, y\_pred\_binary)

Classification\_rep – classification\_report(y\_test, y\_pred\_binary)

Print(f”Accuracy: faccuracy)”)

Print(f”Confusion Matrix:\nfconf\_matrix)”)

Print(f”Classification Report:\n(classification\_rep)”)

✅️0.0s

Accuracy: 0.7337662337662337

Confusion Matrix:

[[76 23]

[18 37]]

Classification Report:

Precision

Recall f1-score

Support

0.81

0.62

0.77

0.67

0.79

0.64

99

55

1

Accuracy

Macro avg

Weighted avg

0.73

0.72

0.74

154

154

154

0.71

0.74

0.72

0.73

***SUPPORT VECTOR CLASSIFIER:***

Svc\_classifier = SVC(kernel=’rbf’, random\_state=42, C=3)

# Train

Svc\_classifier.fit(X\_train, y\_train)

# Predict

Y\_pred = svc\_classifier.predict(X\_test)

# Evaluate

Accuracy = accuracy\_score(y\_test, y\_pred)

Conf\_matrix = confusion\_matrix(y\_test, y\_pred)

Classification \_rep = classification\_ report(y\_test, y\_pred)

Print(f”Accuracy: faccuracy)”)

Print(f”Confusion Matrix:\n/conf\_matrix)”)

Print(f”Classification Report:\n(classification\_rep)”)

✅️0.0s

Accuracy: 0.7727272727272727

Confusion Matrix:

[[86 13]

[22 33]]

Classification Report:

Precision

Recall f1-score

Support

1

8.88

0.72

0.87

0.60

8.83

0.65

99

55

Accuracy

Macro avg

Weighted avg

0.77

0.74

0.77

154

154

154

0.76

0.77

0.73

0.77

***DECISION TREE REGRESSION:***

# Decision Trees

Decision\_tree\_regressor – DecisionTreeRegressor(random\_state=42)

Decision\_tree\_regressor.fit(x\_train, y\_train)

Y\_pred\_dt = decision\_tree\_regressor.predict(x\_test)

Mse\_dt – mean\_squared\_error(y\_test, y\_pred\_dt)

R2\_dt = r2\_score(y\_test, y\_pred\_dt)

Print(“Mean Squared Error: “, mse\_dt)

Print(“R-sqaured: “, r2\_dt)

✅️0.Os

Mean Squared Error: 129792424.45102002

R-sqaured: 0.9243570060021165

Plt.figure(figsize-(8, 6))

Plt.scatter(y\_test, y\_pred\_rf, color=’blue’, label=’Actual’)

Plt.plot(y\_test, y\_test, color=’red`, linestyle=’-‘, linewidth=2, label=’Predicted’)

Plt.title(‘Random Forest Regression’)

Plt.legend()

Plt.grid(True)

Plt.show()

✅️0.1s

Plt.figure(figsize=(8, 6))

Plt.scatter(y, y\_pred\_PR, color-‘blue’, label=”Actual Data (after poly\_features)’)

Plt.plot(y, y, color-‘red`, linestyle-“-‘, linewidth-2, label-‘Predicted Data’)

Plt.title(‘polynomial Regression’)

Plt.legend()

Plt.grid(True)

Plt.show()

✅️0.2s

**PROJECT3:**

***POLYNOMIAL REGRESSION:***

Poly\_features = PolynomialFeatures (degree=2, include\_ bias-False)

X \_poly – poly\_features.fit\_ transform(x)

# Training the Polynomial Regression model

Poly\_regression = LinearRegression()

Poly\_regression.fit(X\_poly, y)

# Making predictions

Y\_pred\_PR = poly\_regression.predict(X\_poly)

# Evaluating the model

Mse = mean squared\_error(y, y\_pred\_PR)

R2= r2\_score(y, y\_pred\_PR)

Print(f”Mean Squared Error: mse:.2f)”)

Print(f”R-squared: (r2:.2f)”)

✅️0.0s

Mean Squared Error: 72642570.85

R-squared: 0.95

***SUPPORT VECTOR REGRESSION:***

Scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(x\_train)

X\_test\_scaled = scaler.transform(x\_test)

# Create and train the SVR model

Svr = SVR(kernel=”poly’, gamma=6)

Svr.fit(x\_train\_scaled, y\_train)

# Predict on the test set

Y\_pred\_svr = svr.predict(x\_test\_scaled)

# alculate R-squared and MSE

R2\_svr= r2\_score(y\_test, y\_pred\_svr)

Mse\_svr= mean\_squared\_error(y\_test, y\_pred\_svr)

Print(f”R-squared (Test set): (r2\_svr:.4f)”)

Print(f”Mean Squared Error (Test set): (mse\_svr:.2f>”)

✅️0.0s

R-squared (Test set): 0.8255

Mean Squared Error (Test set): 299477775.19

Plt.figure(figsize=(6, 6))

Plt.scatter(y\_test, y\_pred\_gb, color-‘blue’, label=’Actual Data’)

Plt.plot(y\_test, y\_test, color-‘red’, linestyle-‘-“, linewidth-2, label-‘Predicted Data’)

Plt.title(‘Gradient Boosting Regression’)

Plt.legend()

Plt.grid(True)

Plt.show()

✅️0.1s

Plt.figure(figsize=(6, 6))

Plt.scatter(df[‘Actual Value’], df[‘Predicted value’], color-‘blue’, label=`Actual Data’)

Plt.plot(df[‘Actual Value’], df[‘Actual Value’], color=”red’, linestyle=’-‘, linewidth-2, label- ‘Predicted Data’)

Plt.title(‘Linear Regression’)

Plt.legend()

Plt.grid(True)

Plt.show()

✅️0.5s

DATASETS :

1. <https://www.kaggle.com/datasets/abhishek14398/50startups> for project1
2. <https://www.kaggle.com/datasets/saurabh00007/diabetescsv> for project2
3. <https://www.kaggle.com/code/alyaratna/dibimbing-data-science-alya> for project3
4. <https://www.kaggle.com/datasets/msambare/fer2013> for project4