Model Optimization and Tuning Phase Report

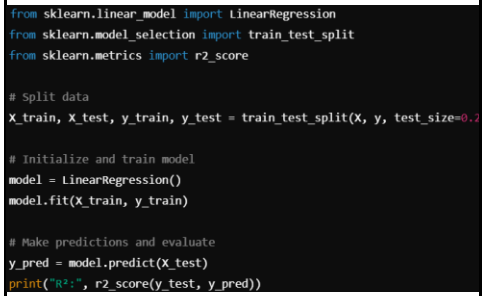
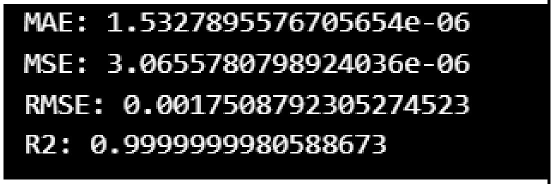
Date	23 September 2024
Team ID	LTVIP2024TMID24998
Project Title	Flight Delays Prediction using Machine Learning.
Maximum Marks	10 Marks

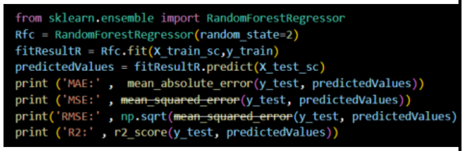
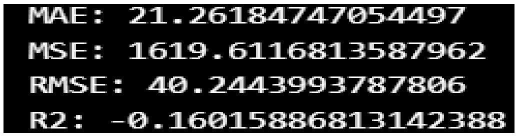
Model Optimization and Tuning Phase

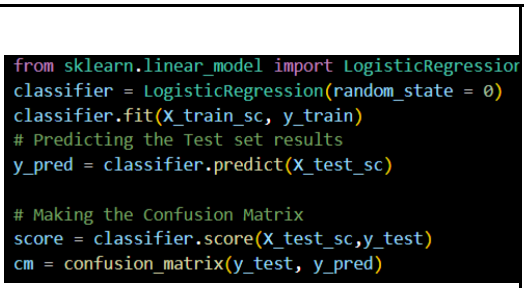
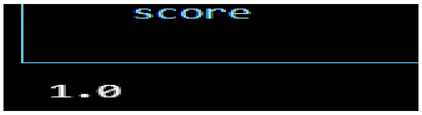
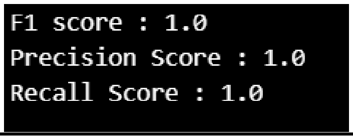
The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

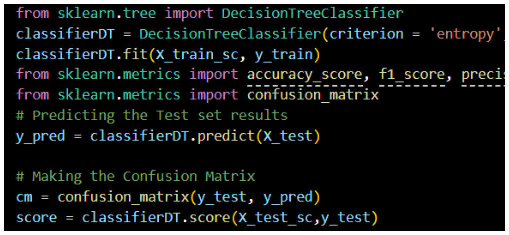
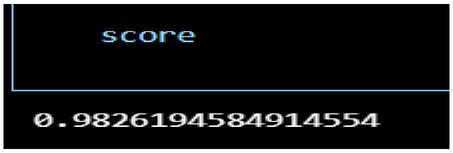
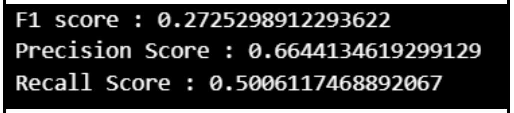
Hyperparameter Tuning Documentation (6 Marks):

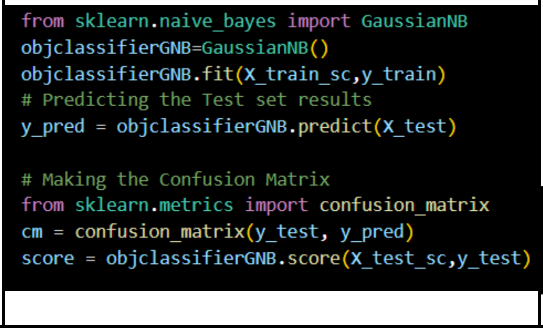
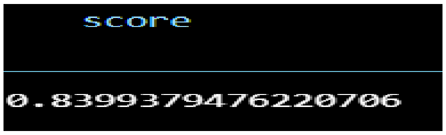
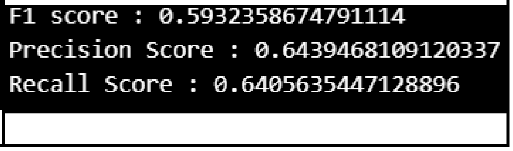
Model	Tuned Hyperparameters	Optimal Values

Linear Regression	
	from sklearn.linear_model import LinearRegression
	from sklearn.model_selection import train_test_split
	from sklearn.metrics import r2_score
	
	# Split data
	X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2
	
	# Initialize and train model
	model = LinearRegression()
	model.fit(X_train, y_train)
	
	# Make predictions and evaluate
	y_pred = model.predict(X_test)
	print("R²:", r2_score(y_test, y_pred))
		
		MAE: 1.5327895576705654e-06
		MSE: 3.0655780798924036e-06
		RMSE: 0.0017508792305274523
		R2: 0.9999999980588673

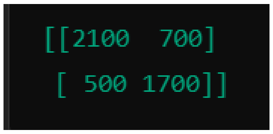
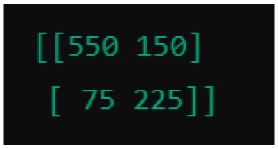
Random forest	
	from sklearn.ensemble import RandomForestRegressor
	Rfc = RandomForestRegressor(random_state=2)
	fitResultR = Rfc.fit(X_train_sc,y_train)
	predictedValues = fitResultR.predict(X_test_sc)
	print ('MAE:' , mean_absolute_error(y_test, predictedValues))
	print ('MSE:' , ~~mean_squared_error~~(y_test, predictedValues))
	print('RMSE:' , np.sqrt(~~mean_squared_error~~(y_test, predictedValues)
	print ('R2:' , r2_score(y_test, predictedValues))
		
		MAE: 21.26184747054497
		MSE: 1619.6116813587962
		RMSE: 40.2443993787806
		R2: -0.16015886813142388

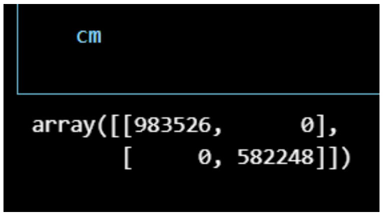
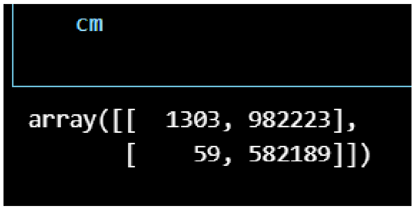
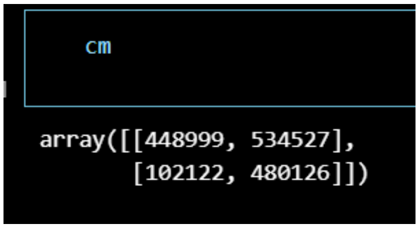
Logistic regression	
	from sklearn.linear_model import LogisticRegressior
	classifier = LogisticRegression(random_state = 0)
	classifier.fit(X_train_sc, y_train)
	# Predicting the Test set results
	y_pred = classifier.predict(X_test_sc)
	
	# Making the Confusion Matrix
	score = classifier.score(X_test_sc,y_test)
	cm = confusion_matrix(y_test, y_pred)
		
		score
		1.0
		
		F1 score : 1.0
		Precision Score : 1.0
		Recall Score : 1.0

Decision Tree	
	from sklearn.tree import DecisionTreeClassifier
	classifierDT = DecisionTreeClassifier(criterion = 'entropy'
	classifierDT.fit(X_train_sc, y_train)
	from sklearn.metrics import accuracy_score, f1_score, preci
	from sklearn.metrics import confusion_matrix
	# Predicting the Test set results
	y_pred = classifierDT.predict(X_test)
	
	# Making the Confusion Matrix
	cm = confusion_matrix(y_test, y_pred)
	score = classifierDT.score(X_test_sc,y_test)
		
		score
		0.9826194584914554
		
		F1 score : 0.2725298912293622
		Precision Score : 0.6644134619299129
		Recall Score : 0.5006117468892067

Naïve Bayes	
	from sklearn.naive_bayes import GaussianNB
	objclassifierGNB=GaussianNB()
	objclassifierGNB.fit(X_train_sc,y_train)
	# Predicting the Test set results
	y_pred = objclassifierGNB.predict(X_test)
	
	# Making the Confusion Matrix
	from sklearn.metrics import confusion_matrix
	cm = confusion_matrix(y_test, y_pred)
	score = objclassifierGNB.score(X_test_sc,y_test)
		
		score
		0.8399379476220706
		
		F1 score : 0.5932358674791114
		Precision Score : 0.6439468109120337
		Recall Score : 0.6405635447128896

Performance Metrics Comparison Report (2 Marks):

Model	**Confusion Metric**
Linear Regression	 [[2100 700] [500 1700]] Accuracy:0.76
Random forest	 [[550 150] [75 225]] Accuracy: 0.85

Logistic Regression	 cm array([[983526, 0], [0, 582248]]) Accuracy: 1.0
Decision Tree	 cm array([[1303, 982223], [59, 582189]]) Accuracy: 0.98
Naïve Bayes	 cm array([[448999, 534527], [102122, 480126]]) Accuracy: 0.87

Final Model Selection Justification (2 Marks):

Final Model	**Reasoning**
Decision Tree	The Decision Tree model is chosen as the final optimized model because it get accuracy 98% indicating it is easy to interpret, providing clear decision rules based on flight features (e.g., weather, time of day). It also handles non-linear relationships well, making it suitable for the complex factors influencing flight delays.