SMARTBRIDGE
Let's Bridge the Gap
a Veranda Enterprise

Smart Internz

Model Optimization and Tuning Phase Template

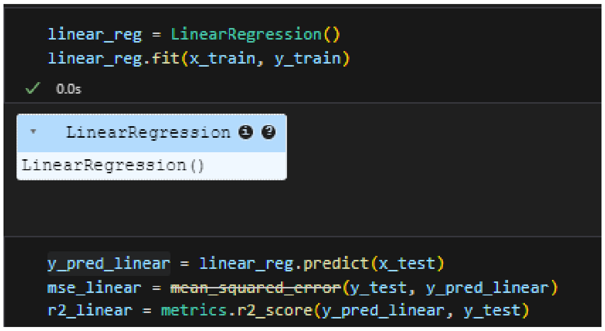
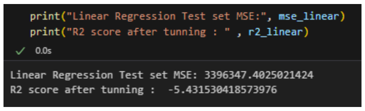
Date	16th October 2024
Team ID	LTVIP2024TMID24968
Project Title	TrafficTelligence Advanced Traffic Volume Estimation with Machine Learning
Maximum Marks	10 Marks

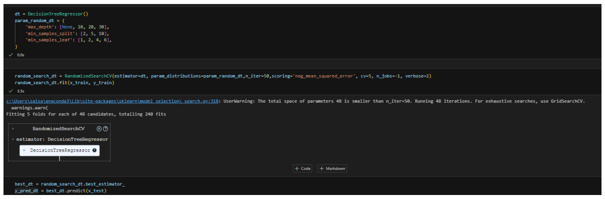
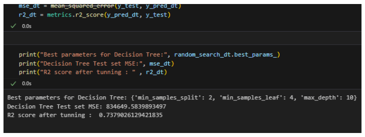
Model Optimization and Tuning Phase

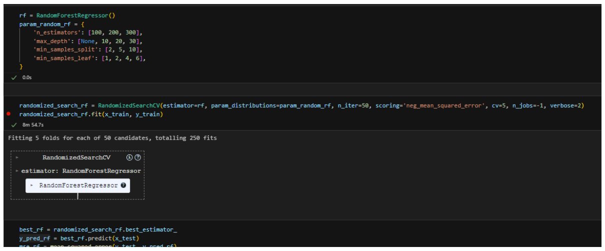
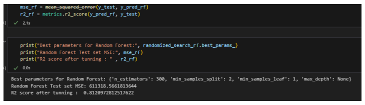
The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

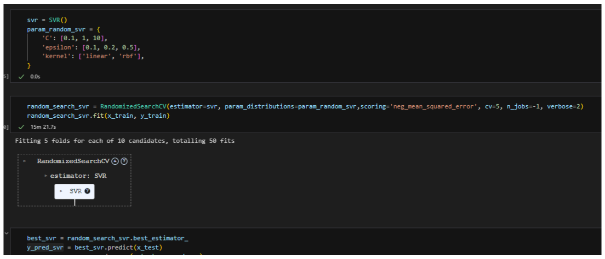
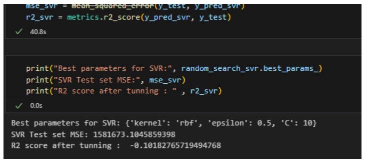
Hyperparameter Tuning Documentation (6 Marks):

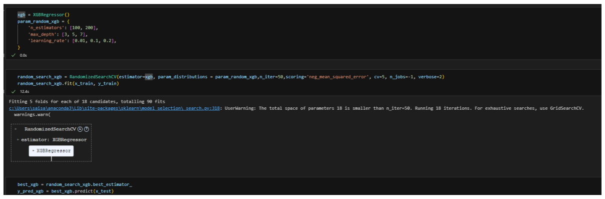
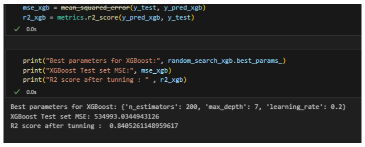
Model	Tuned Hyperparameters	Optimal Values

Linear Regression	
	linear_reg = LinearRegression()
	linear_reg.fit(x_train, y_train)
	✓ 0.0s
	LinearRegression
	LinearRegression()
	y_pred_linear = linear_reg.predict(x_test)
	mse_linear = ~~mean_squared_error~~(y_test, y_pred_linear)
	r2_linear = metrics.r2_score(y_pred_linear, y_test)
		
		print("Linear Regression Test set MSE:", mse_linear)
		print("R2 score after tunning : " , r2_linear)
		✓ 0.0s
		Linear Regression Test set MSE: 3396347.4025021424
		R2 score after tunning : -5.431530418573976

Decision Tree Regressor	
	dt = DecisionTreeRegressor()
	param_random_dt = {
	'max_depth': [None, 10, 20, 30],
	'min_samples_split': [2, 5, 10],
	'min_samples_leaf': [1, 2, 4, 6],
	}
	✓ 0.0s
	random_search_dt = RandomizedSearchCV(estimator=dt, param_distributions=param_random_dt,n_iter=50,scoring='neg_mean_squared_error', cv=5, n_jobs=-1, verbose=2)
	random_search_dt.fit(x_train, y_train)
	✓ 5.5s
	c:\Users\saisa\anaconda3\Lib\site-packages\sklearn\model_selection_search.py:318: UserWarning: The total space of parameters 48 is smaller than n_iter=50. Running 48 iterations. For exhaustive searches, use GridSearchCV.
	warnings.warn(
	Fitting 5 folds for each of 48 candidates, totalling 240 fits
	RandomizedSearchCV
	estimator: DecisionTreeRegressor
	DecisionTreeRegressor
	+ Code + Markdown
	best_dt = random_search_dt.best_estimator_
	y_pred_dt = best_dt.predict(x_test)
		
		mse_dt = ~~mean_squared_error~~(y_test, y_pred_dt)
		r2_dt = metrics.r2_score(y_pred_dt, y_test)
		✓ 0.0s
		print("Best parameters for Decision Tree:", random_search_dt.best_params_)
		print("Decision Tree Test set MSE:", mse_dt)
		print("R2 score after tunning : " , r2_dt)
		✓ 0.0s
		Best parameters for Decision Tree: {'min_samples_split': 2, 'min_samples_leaf': 4, 'max_depth': 10}
		Decision Tree Test set MSE: 834649.5839893497
		R2 score after tunning : 0.7379026129421835

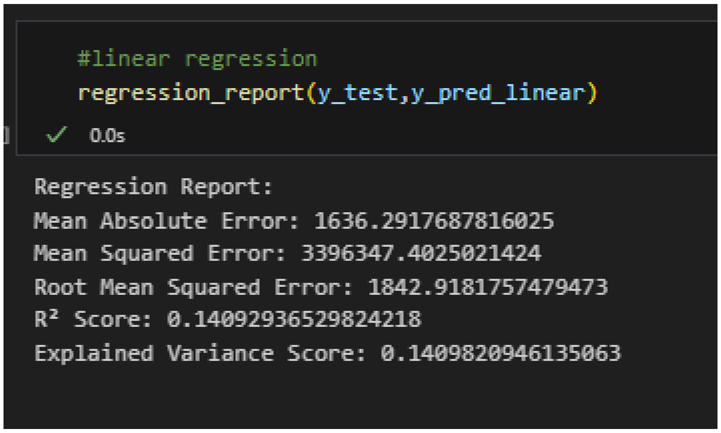
Random Forest
Regressor
	
	rf = RandomForestRegressor()
	param_random_rf = {
	'n_estimators': [100, 200, 300],
	'max_depth': [None, 10, 20, 30],
	'min_samples_split': [2, 5, 10],
	'min_samples_leaf': [1, 2, 4, 6],
	}
	✓ 0.0s
	randomized_search_rf = RandomizedSearchCV(estimator=rf, param_distributions=param_random_rf, n_iter=50, scoring='neg_mean_squared_error', cv=5, n_jobs=-1, verbose=2)
	randomized_search_rf.fit(x_train, y_train)
	✓ 8m 54.7s
	Fitting 5 folds for each of 50 candidates, totalling 250 fits
	RandomizedSearchCV
	estimator: RandomForestRegressor
	RandomForestRegressor
	best_rf = randomized_search_rf.best_estimator_
	y_pred_rf = best_rf.predict(x_test)
	mse_rf = mean_squared_error(y_test, y_pred_rf)
		
		mse_rf = ~~mean_squared_error~~(y_test, y_pred_rf)
		r2_rf = metrics.r2_score(y_pred_rf, y_test)
		✓ 2.1s
		print("Best parameters for Random Forest:", randomized_search_rf.best_params_)
		print("Random Forest Test set MSE:", mse_rf)
		print("R2 score after tunning : " , r2_rf)
		✓ 0.0s
		Best parameters for Random Forest: {'n_estimators': 300, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_depth': None}
		Random Forest Test set MSE: 611318.5661813644
		R2 score after tunning : 0.8120972812517622

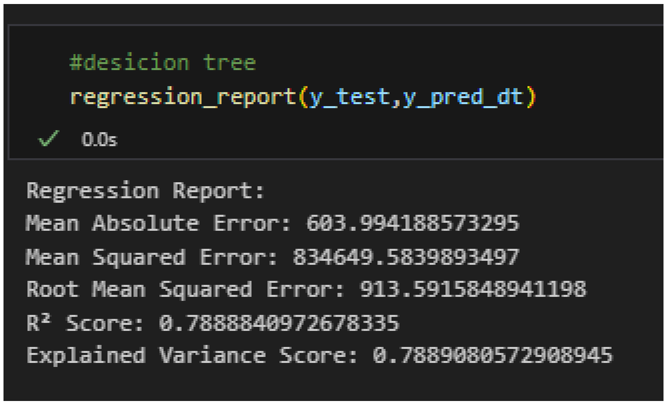
SVR	
	svr = SVR()
	param_random_svr = {
	'C': [0.1, 1, 10],
	'epsilon': [0.1, 0.2, 0.5],
	'kernel': ['linear', 'rbf'],
	}
	✓ 0.0s
	random_search_svr = RandomizedSearchCV(estimator=svr, param_distributions=param_random_svr,scoring='neg_mean_squared_error', cv=5, n_jobs=-1, verbose=2)
	random_search_svr.fit(x_train, y_train)
	✓ 15m 21.7s
	Fitting 5 folds for each of 10 candidates, totalling 50 fits
	RandomizedSearchCV
	estimator: SVR
	SVR
	best_svr = random_search_svr.best_estimator_
	y_pred_svr = best_svr.predict(x_test)
		
		mse_svr = ~~mean_squared_error~~(y_test, y_pred_svr)
		r2_svr = metrics.r2_score(y_pred_svr, y_test)
		✓ 40.8s
		print("Best parameters for SVR:", random_search_svr.best_params_)
		print("SVR Test set MSE:", mse_svr)
		print("R2 score after tunning : " , r2_svr)
		✓ 0.0s
		Best parameters for SVR: {'kernel': 'rbf', 'epsilon': 0.5, 'C': 10}
		SVR Test set MSE: 1581673.1045859398
		R2 score after tunning : -0.10182765719494768

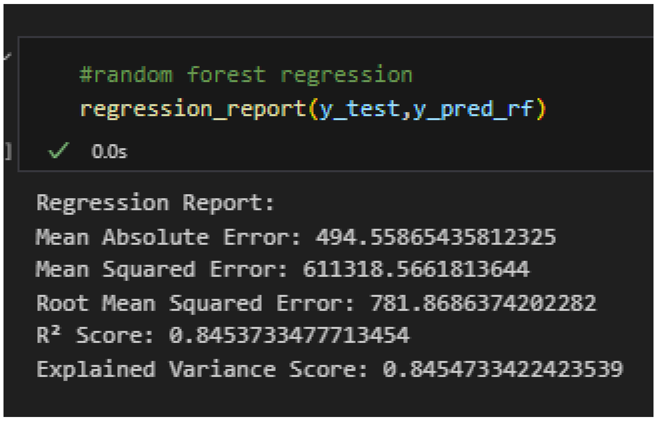
XGBoost	
	xgb = XGBRegressor()
	param_random_xgb = {
	'n_estimators': [100, 200],
	'max_depth': [3, 5, 7],
	'learning_rate': [0.01, 0.1, 0.2],
	}
	✓ 0.0s
	random_search_xgb = RandomizedSearchCV(estimator=xgb, param_distributions = param_random_xgb,n_iter=50,scoring='neg_mean_squared_error', cv=5, n_jobs=-1, verbose=2)
	random_search_xgb.fit(x_train, y_train)
	✓ 12.4s
	Fitting 5 folds for each of 18 candidates, totalling 90 fits
	c:\Users\saisa\anaconda3\Lib\site-packages\sklearn\model_selection_search.py:318: UserWarning: The total space of parameters 18 is smaller than n_iter=50. Running 18 iterations. For exhaustive searches, use GridSearchCV.
	warnings.warn(
	RandomizedSearchCV
	estimator: XGBRegressor
	XGBRegressor
	best_xgb = random_search_xgb.best_estimator_
	y_pred_xgb = best_xgb.predict(x_test)
		
		mse_xgb = ~~mean_squared_error~~(y_test, y_pred_xgb)
		r2_xgb = metrics.r2_score(y_pred_xgb, y_test)
		✓ 0.0s
		print("Best parameters for XGBoost:", random_search_xgb.best_params_)
		print("XGBoost Test set MSE:", mse_xgb)
		print("R2 score after tunning : " , r2_xgb)
		✓ 0.0s
		Best parameters for XGBoost: {'n_estimators': 200, 'max_depth': 7, 'learning_rate': 0.2}
		XGBoost Test set MSE: 534993.0344943126
		R2 score after tunning : 0.8405261148959617

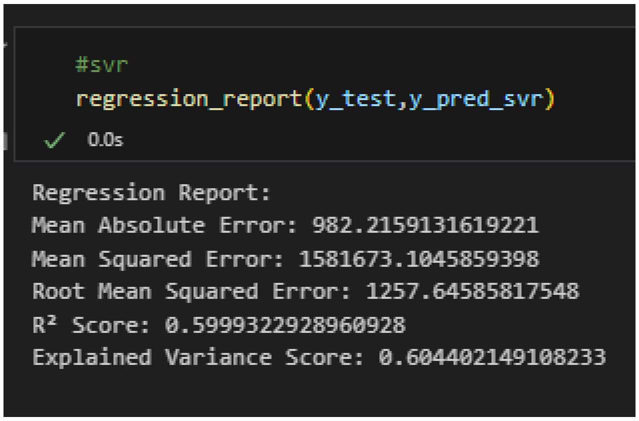
Performance Metrics Comparison Report (2 Marks):

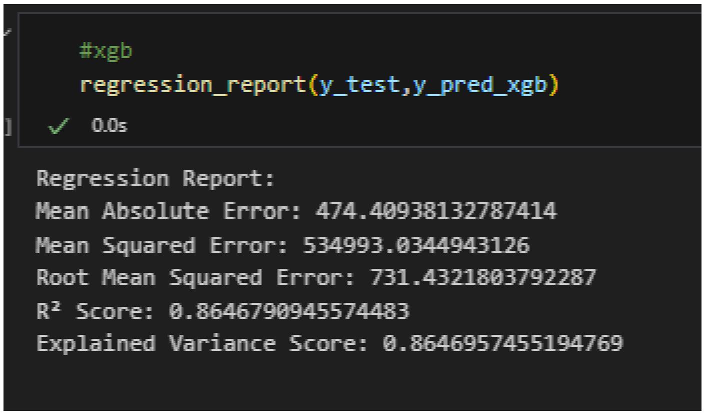
Model	Optimized Metric

Linear Regression	
	#linear regression
	regression_report(y_test,y_pred_linear)
	✓ 0.0s
	Regression Report:
	Mean Absolute Error: 1636.2917687816025
	Mean Squared Error: 3396347.4025021424
	Root Mean Squared Error: 1842.9181757479473
	R² Score: 0.14092936529824218
	Explained Variance Score: 0.1409820946135063

Decision Tree
Regressor
	
	#desicion tree
	regression_report(y_test,y_pred_dt)
	✓ 0.0s
	Regression Report:
	Mean Absolute Error: 603.994188573295
	Mean Squared Error: 834649.5839893497
	Root Mean Squared Error: 913.5915848941198
	R² Score: 0.7888840972678335
	Explained Variance Score: 0.7889080572908945

Random Forest
Regressor
	
	#random forest regression
	regression_report(y_test,y_pred_rf)
	✓ 0.0s
	Regression Report:
	Mean Absolute Error: 494.55865435812325
	Mean Squared Error: 611318.5661813644
	Root Mean Squared Error: 781.8686374202282
	R² Score: 0.8453733477713454
	Explained Variance Score: 0.8454733422423539

SVR	
	#svr
	regression_report(y_test,y_pred_svr)
	✓ 0.0s
	Regression Report:
	Mean Absolute Error: 982.2159131619221
	Mean Squared Error: 1581673.1045859398
	Root Mean Squared Error: 1257.64585817548
	R² Score: 0.5999322928960928
	Explained Variance Score: 0.604402149108233

XGBoost	
	#xgb
	regression_report(y_test,y_pred_xgb)
	✓ 0.0s
	Regression Report:
	Mean Absolute Error: 474.40938132787414
	Mean Squared Error: 534993.0344943126
	Root Mean Squared Error: 731.4321803792287
	R² Score: 0.8646790945574483
	Explained Variance Score: 0.8646957455194769

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Extreme Gradient Boosting	XGBoost was chosen as the final optimized model due to its high predictive accuracy and efficiency. Its built-in regularization helped prevent overfitting, while its ability to handle missing values simplified preprocessing. Additionally, XGBoost provides valuable insights into feature importance, enhancing interpretability and model refinement to align with project objectives. justifying it as a final model