Model Development Phase Template

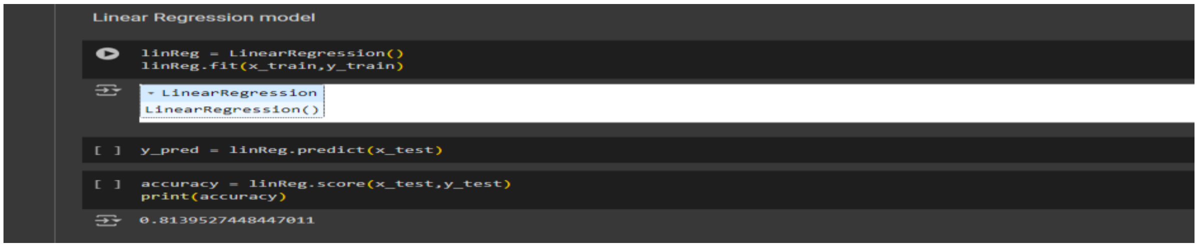
Date	07 July 2024
Team ID	team-739715
Project Title	House Rent Price Prediction Using Machine Learning
Maximum Marks	4 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

Initial Model Training Code:

Linear Regression Model



Linear Regression model

```
linReg = LinearRegression()
linReg.fit(x_train,y_train)
```

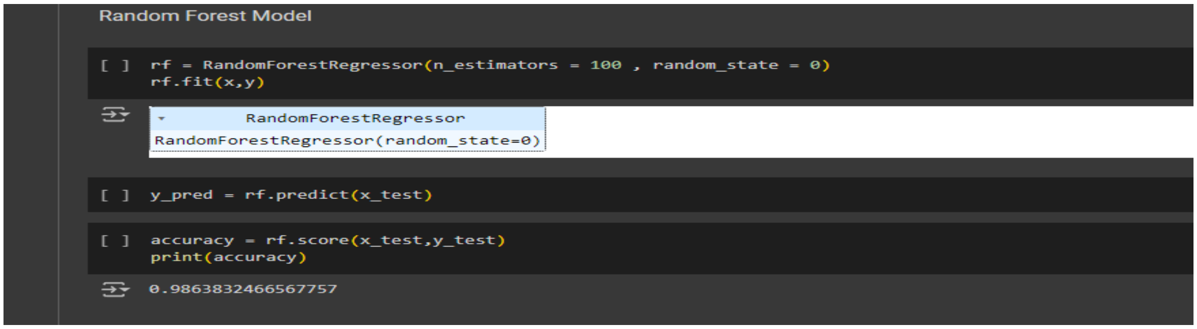
```
LinearRegression
LinearRegression()
```

```
y_pred = linReg.predict(x_test)
```

```
accuracy = linReg.score(x_test,y_test)
print(accuracy)
```

```
0.8139527448447011
```

Random Forest Model



Random Forest Model

```
rf = RandomForestRegressor(n_estimators = 100 , random_state = 0)
rf.fit(x,y)
```

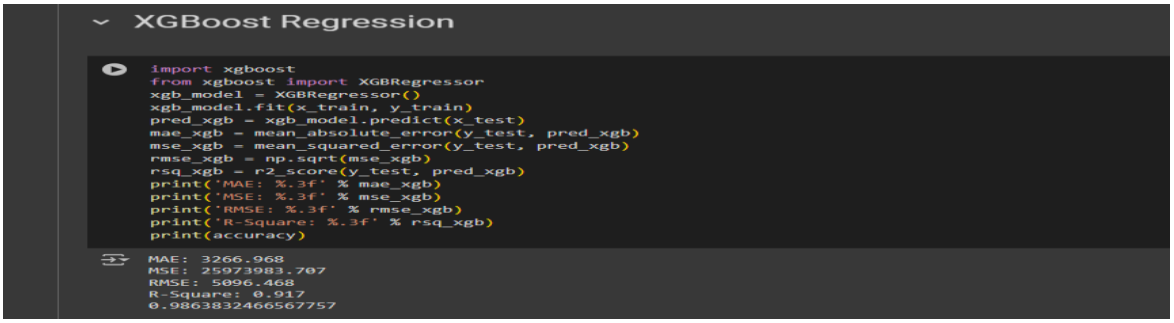
```
RandomForestRegressor
RandomForestRegressor(random_state=0)
```

```
y_pred = rf.predict(x_test)
```

```
accuracy = rf.score(x_test,y_test)
print(accuracy)
```

```
0.9863832466567757
```

XGBoost Regression Model

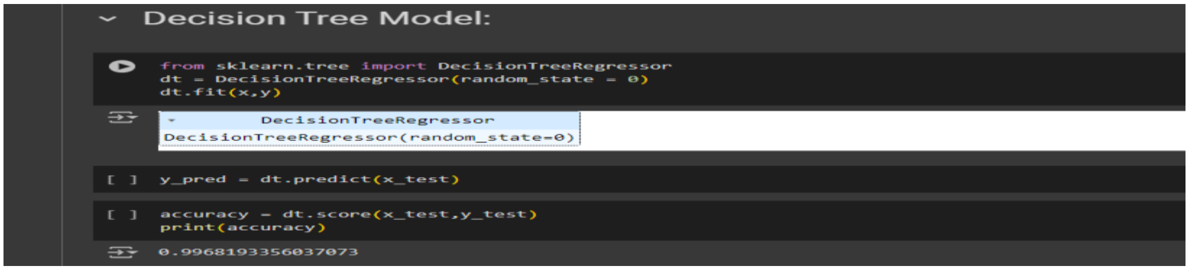


XGBoost Regression

```
import xgboost
from xgboost import XGBRegressor
xgb_model = XGBRegressor()
xgb_model.fit(x_train, y_train)
pred_xgb = xgb_model.predict(x_test)
mae_xgb = mean_absolute_error(y_test, pred_xgb)
mse_xgb = mean_squared_error(y_test, pred_xgb)
rmse_xgb = np.sqrt(mse_xgb)
rsq_xgb = r2_score(y_test, pred_xgb)
print('MAE: %.3f' % mae_xgb)
print('MSE: %.3f' % mse_xgb)
print('RMSE: %.3f' % rmse_xgb)
print('R-Square: %.3f' % rsq_xgb)
print(accuracy)
```

```
MAE: 3266.968
MSE: 25973983.707
RMSE: 5096.468
R-Square: 0.917
0.9863832466567757
```

Decision Tree Model



Decision Tree Model:

```
from sklearn.tree import DecisionTreeRegressor
dt = DecisionTreeRegressor(random_state = 0)
dt.fit(x,y)
```

```
DecisionTreeRegressor
DecisionTreeRegressor(random_state=0)
```

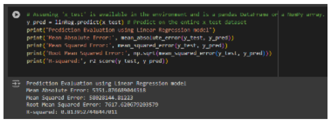
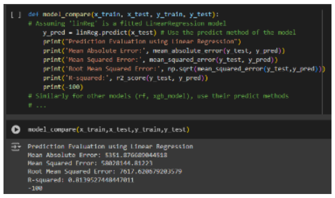
```
[ ] y_pred = dt.predict(x_test)
```

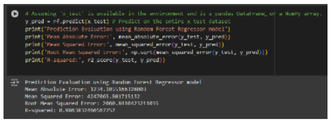
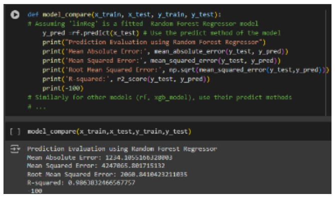
```
[ ] accuracy = dt.score(x_test,y_test)
    print(accuracy)
```

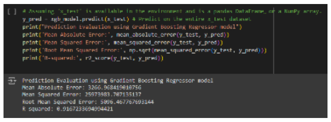
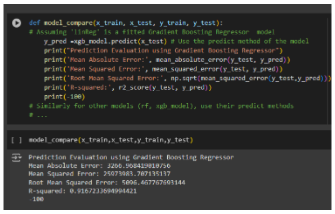
```
0.9968193356037073
```

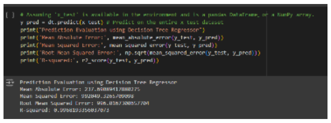
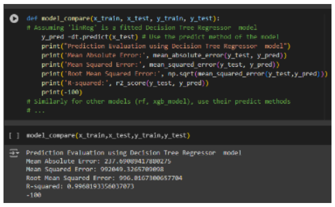
Model Validation and Evaluation Report:

Model	Regression Report	Accuracy	Regression Matrix

Linear Regression	
	`# Assuming 'x_test' is available in the environment and is a pandas DataFrame or a NumPy array.`
	`y_pred = linReg.predict(x_test) # Predict on the entire x test dataset`
	`print("Prediction Evaluation using Linear Regression model")`
	`print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred))`
	`print('Mean Squared Error:', mean_squared_error(y_test, y_pred))`
	`print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred)))`
	`print('R-squared:', r2_score(y_test, y_pred))`
	`Prediction Evaluation using Linear Regression model`
	`Mean Absolute Error: 5351.876689044518`
	`Mean Squared Error: 58028144.81223`
	`Root Mean Squared Error: 7617.620679203579`
	`R-squared: 0.8139527448447011`
		81.3%	
			`[ ] def model_compare(x_train, x_test, y_train, y_test):`
			`# Assuming 'linReg' is a fitted LinearRegression model`
			`    y_pred = linReg.predict(x_test) # Use the predict method of the model`
			`    print("Prediction Evaluation using Linear Regression")`
			`    print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred))`
			`    print('Mean Squared Error:', mean_squared_error(y_test, y_pred))`
			`    print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test,y_pred)))`
			`    print('R-squared:', r2_score(y_test, y_pred))`
			`    print(-100)`
			`# Similarly for other models (rf, xgb_model), use their predict methods`
			`# ...`
			`model_compare(x_train,x_test,y_train,y_test)`
			`Prediction Evaluation using Linear Regression`
			`Mean Absolute Error: 5351.876689044518`
			`Mean Squared Error: 58028144.81223`
			`Root Mean Squared Error: 7617.620679203579`
			`R-squared: 0.8139527448447011`
			`-100`

Random Forest Regressor	
	`# Assuming 'x_test' is available in the environment and is a pandas DataFrame or a NumPy array.`
	`y_pred = rf.predict(x_test) # Predict on the entire x test dataset`
	`print("Prediction Evaluation using Random Forest Regressor model")`
	`print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred))`
	`print('Mean Squared Error:', mean_squared_error(y_test, y_pred))`
	`print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred)))`
	`print('R-squared:', r2_score(y_test, y_pred))`
	`Prediction Evaluation using Random Forest Regressor model`
	`Mean Absolute Error: 1234.1055166328003`
	`Mean Squared Error: 4247865.881715132`
	`Root Mean Squared Error: 2060.8410423211035`
	`R-squared: 0.9863832466567757`
		98.6%	
			`def model_compare(x_train, x_test, y_train, y_test):`
			`# Assuming 'linReg' is a fitted  Random Forest Regressor model`
			`    y_pred =rf.predict(x_test) # Use the predict method of the model`
			`    print("Prediction Evaluation using Random Forest Regressor")`
			`    print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred))`
			`    print('Mean Squared Error:', mean_squared_error(y_test, y_pred))`
			`    print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test,y_pred)))`
			`    print('R-squared:', r2_score(y_test, y_pred))`
			`    print(-100)`
			`# Similarly for other models (rf, xgb_model), use their predict methods`
			`# ...`
			`[ ] model_compare(x_train,x_test,y_train,y_test)`
			`Prediction Evaluation using Random Forest Regressor`
			`Mean Absolute Error: 1234.1055166328003`
			`Mean Squared Error: 4247865.881715132`
			`Root Mean Squared Error: 2060.8410423211035`
			`R-squared: 0.9863832466567757`
			`-100`

XGBoost Regression	
	`[ ] # Assuming 'x_test' is available in the environment and is a pandas DataFrame, or a NumPy array.`
	`y_pred = xgb_model.predict(x_test) # Predict on the entire x_test dataset`
	`print("Prediction Evaluation using Gradient Boosting Regressor model")`
	`print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred))`
	`print('Mean Squared Error:', mean_squared_error(y_test, y_pred))`
	`print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred)))`
	`print('R-squared:', r2_score(y_test, y_pred))`
	`Prediction Evaluation using Gradient Boosting Regressor model`
	`Mean Absolute Error: 3266.968419010756`
	`Mean Squared Error: 25973983.707135137`
	`Root Mean Squared Error: 5096.467767693144`
	`R squared: 0.9167233694994421`
		91.6%	
			`def model_compare(x_train, x_test, y_train, y_test):`
			`# Assuming 'linReg' is a fitted Gradient Boosting Regressor  model`
			`    y_pred =xgb_model.predict(x_test) # Use the predict method of the model`
			`    print("Prediction Evaluation using Gradient Boosting Regressor")`
			`    print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred))`
			`    print('Mean Squared Error:', mean_squared_error(y_test, y_pred))`
			`    print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test,y_pred)))`
			`    print('R-squared:', r2_score(y_test, y_pred))`
			`    print(-100)`
			`# Similarly for other models (rf, xgb_model), use their predict methods`
			`# ...`
			`[ ] model_compare(x_train,x_test,y_train,y_test)`
			`Prediction Evaluation using Gradient Boosting Regressor`
			`Mean Absolute Error: 3266.968419010756`
			`Mean Squared Error: 25973983.707135137`
			`Root Mean Squared Error: 5096.467767693144`
			`R-squared: 0.9167233694994421`
			`-100`

Decision Tree	
	[ ] # Assuming 'x_test' is available in the environment and is a pandas DataFrame, or a NumPy array.
	y pred = dt.predict(x test) # Predict on the entire x test dataset
	print("Prediction Evaluation using Decision Tree Regressor")
	print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred))
	print('Mean Squared Error:', mean squared error(y test, y pred))
	print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred)))
	print('R-squared:', r2_score(y_test, y_pred))
	Prediction Evaluation using Decision Tree Regressor
	Mean Absolute Error: 237.6908941788275
	Mean Squared Error: 992049.3265709098
	Root Mean Squared Error: 996.0167300657704
	R-squared: 0.9960193356037073
		99.6%	
			def model_compare(x_train, x_test, y_train, y_test):
			# Assuming 'linReg' is a fitted Decision Tree Regressor model
			    y_pred -dt.predict(x_test) # Use the predict method of the model
			    print("Prediction Evaluation using Decision Tree Regressor model")
			    print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred))
			    print('Mean Squared Error:', mean_squared_error(y_test, y_pred))
			    print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test,y_pred)))
			    print('R-squared:', r2_score(y_test, y_pred))
			    print(-100)
			# Similarly for other models (rf, xgb_model), use their predict methods
			# ...
			[ ] model_compare(x_train,x_test,y_train,y_test)
			Prediction Evaluation using Decision Tree Regressor model
			Mean Absolute Error: 237.6908941788275
			Mean Squared Error: 992049.3265789098
			Root Mean Squared Error: 996.0167300657704
			R-squared: 0.9960193356037073
			-100