

## Model Development Phase Template

Date	07 July 2024
Team ID	team-740082
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)

[ ] 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)

[ ] 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	<pre> # Importing the libraries import numpy as np from sklearn.linear_model import LinearRegression  # Splitting the dataset into the training set and the test set X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)  # Fitting the Linear Regression model to the training set regressor = LinearRegression() regressor.fit(X_train, y_train)  # Predicting the test set results y_pred = regressor.predict(X_test)  # Evaluating the model performance 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)) </pre>	81.3%	<pre> def model_compare(x_train, x_test, y_train, y_test):     # Importing the libraries     from sklearn.linear_model import LinearRegression      # Fitting the Linear Regression model to the training set     regressor = LinearRegression()     regressor.fit(x_train, y_train)      # Predicting the test set results     y_pred = regressor.predict(x_test)      # Evaluating the model performance     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))      # Similarly for other models (if, xgb_model), use their predict methods     # ...      # Finally, compare the models     model_compare(x_train, x_test, y_train, y_test) </pre>
Random Forest Regressor	<pre> # Importing the libraries import numpy as np from sklearn.ensemble import RandomForestRegressor  # Splitting the dataset into the training set and the test set X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)  # Fitting the Random Forest Regressor model to the training set regressor = RandomForestRegressor() regressor.fit(X_train, y_train)  # Predicting the test set results y_pred = regressor.predict(X_test)  # Evaluating the model performance 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)) </pre>	98.6%	<pre> def model_compare(x_train, x_test, y_train, y_test):     # Importing the libraries     from sklearn.ensemble import RandomForestRegressor      # Fitting the Random Forest Regressor model to the training set     regressor = RandomForestRegressor()     regressor.fit(x_train, y_train)      # Predicting the test set results     y_pred = regressor.predict(x_test)      # Evaluating the model performance     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))      # Similarly for other models (if, xgb_model), use their predict methods     # ...      # Finally, compare the models     model_compare(x_train, x_test, y_train, y_test) </pre>
XGBoost Regression	<pre> # Importing the libraries import numpy as np from sklearn.ensemble import GradientBoostingRegressor  # Splitting the dataset into the training set and the test set X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)  # Fitting the Gradient Boosting Regressor model to the training set regressor = GradientBoostingRegressor() regressor.fit(X_train, y_train)  # Predicting the test set results y_pred = regressor.predict(X_test)  # Evaluating the model performance 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)) </pre>	91.6%	<pre> def model_compare(x_train, x_test, y_train, y_test):     # Importing the libraries     from sklearn.ensemble import GradientBoostingRegressor      # Fitting the Gradient Boosting Regressor model to the training set     regressor = GradientBoostingRegressor()     regressor.fit(x_train, y_train)      # Predicting the test set results     y_pred = regressor.predict(x_test)      # Evaluating the model performance     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))      # Similarly for other models (if, xgb_model), use their predict methods     # ...      # Finally, compare the models     model_compare(x_train, x_test, y_train, y_test) </pre>

## Decision Tree

## 99.6%

```

# Decision Tree is available in the environment and it is a part of sklearn
# And it is available in the environment and it is a part of sklearn

# Import Decision Tree using Decision Tree Regressor
from sklearn.tree import DecisionTreeRegressor

# Import train_test_split using train_test_split
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Create a Decision Tree Regressor
dt = DecisionTreeRegressor()

# Fit the model
dt.fit(X_train, y_train)

# Predict the output
y_pred = dt.predict(X_test)

# Print the results
print("Mean Absolute Error: ", mean_absolute_error(y_test, y_pred))
print("Mean Squared Error: ", mean_squared_error(y_test, y_pred))
print("R-squared: ", r2_score(y_test, y_pred))

```

```

# Predict the output using Decision Tree Regressor
from sklearn.tree import DecisionTreeRegressor

# Import train_test_split using train_test_split
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Create a Decision Tree Regressor
dt = DecisionTreeRegressor()

# Fit the model
dt.fit(X_train, y_train)

# Predict the output
y_pred = dt.predict(X_test)

# Print the results
print("Mean Absolute Error: ", mean_absolute_error(y_test, y_pred))
print("Mean Squared Error: ", mean_squared_error(y_test, y_pred))
print("R-squared: ", r2_score(y_test, y_pred))

```

## 99.6%

```

# Import Decision Tree using Decision Tree Regressor
from sklearn.tree import DecisionTreeRegressor

# Import train_test_split using train_test_split
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Create a Decision Tree Regressor
dt = DecisionTreeRegressor()

# Fit the model
dt.fit(X_train, y_train)

# Predict the output
y_pred = dt.predict(X_test)

# Print the results
print("Mean Absolute Error: ", mean_absolute_error(y_test, y_pred))
print("Mean Squared Error: ", mean_squared_error(y_test, y_pred))
print("R-squared: ", r2_score(y_test, y_pred))

```