Model Development Phase Template

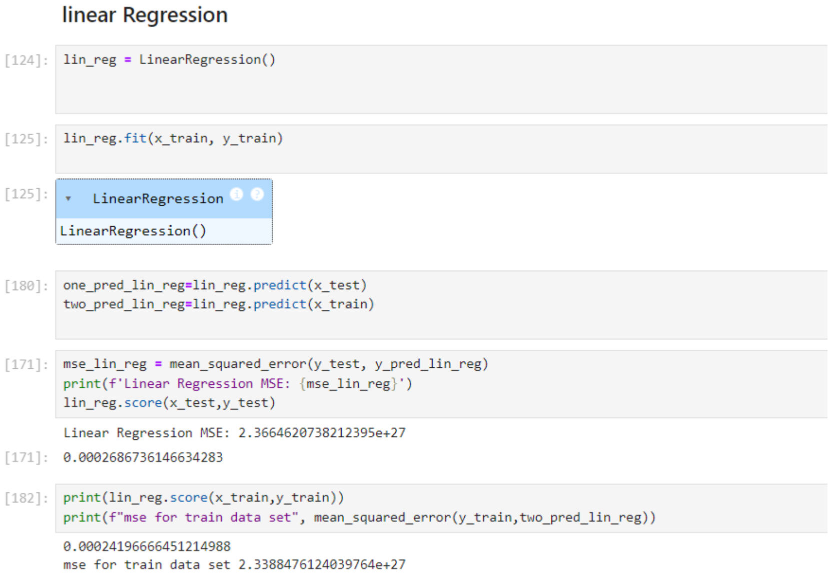
Date	10 JULY 2024
Team ID	SWTID1720000556
Project Title	Predicting Co2 Emission By Countries Using Machine Learning
Maximum Marks	4 Marks

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:

1)



linear Regression

```
[124]: lin_reg = LinearRegression()
```

```
[125]: lin_reg.fit(x_train, y_train)
```

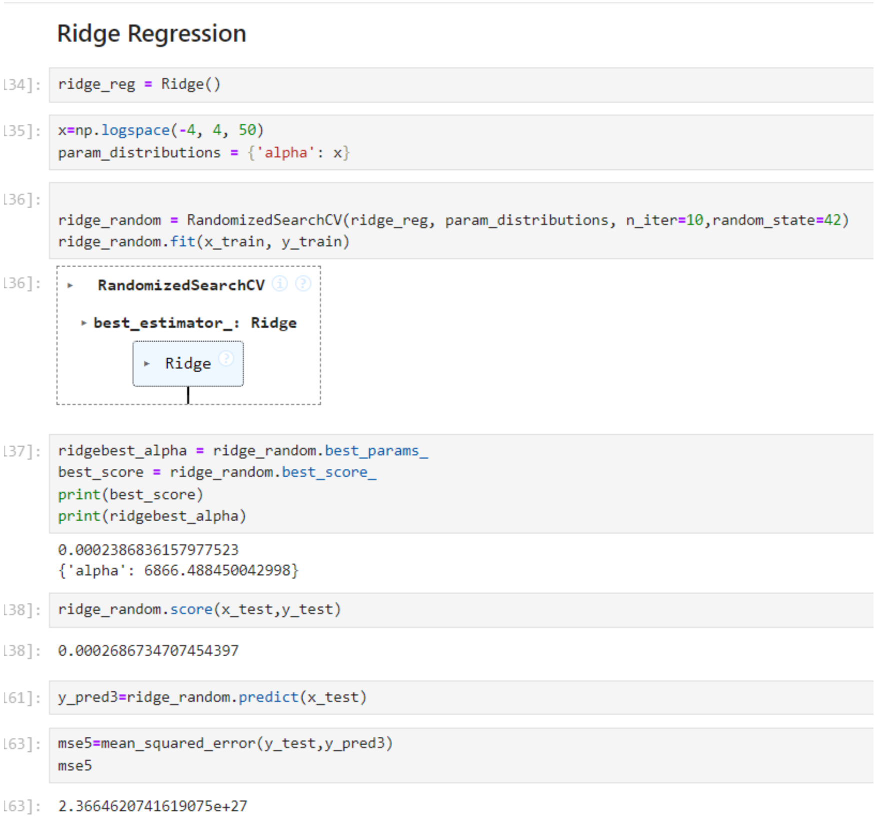
```
[125]: LinearRegression
        LinearRegression()
```

```
[180]: one_pred_lin_reg=lin_reg.predict(x_test)
       two_pred_lin_reg=lin_reg.predict(x_train)
```

```
[171]: mse_lin_reg = mean_squared_error(y_test, y_pred_lin_reg)
       print(f'Linear Regression MSE: {mse_lin_reg}')
       lin_reg.score(x_test,y_test)

       Linear Regression MSE: 2.3664620738212395e+27
[171]: 0.0002686736146634283
```

```
[182]: print(lin_reg.score(x_train,y_train))
       print(f"mse for train data set", mean_squared_error(y_train,two_pred_lin_reg))

       0.00024196666451214988
       mse for train data set 2.3388476124039764e+27
```

2)



Ridge Regression

```
[34]: ridge_reg = Ridge()
```

```
[35]: x=np.logspace(-4, 4, 50)
      param_distributions = {'alpha': x}
```

```
[36]:
      ridge_random = RandomizedSearchCV(ridge_reg, param_distributions, n_iter=10,random_state=42)
      ridge_random.fit(x_train, y_train)
```

```
[36]: ▸ RandomizedSearchCV
        ▸ best_estimator_: Ridge
            ▸ Ridge
```

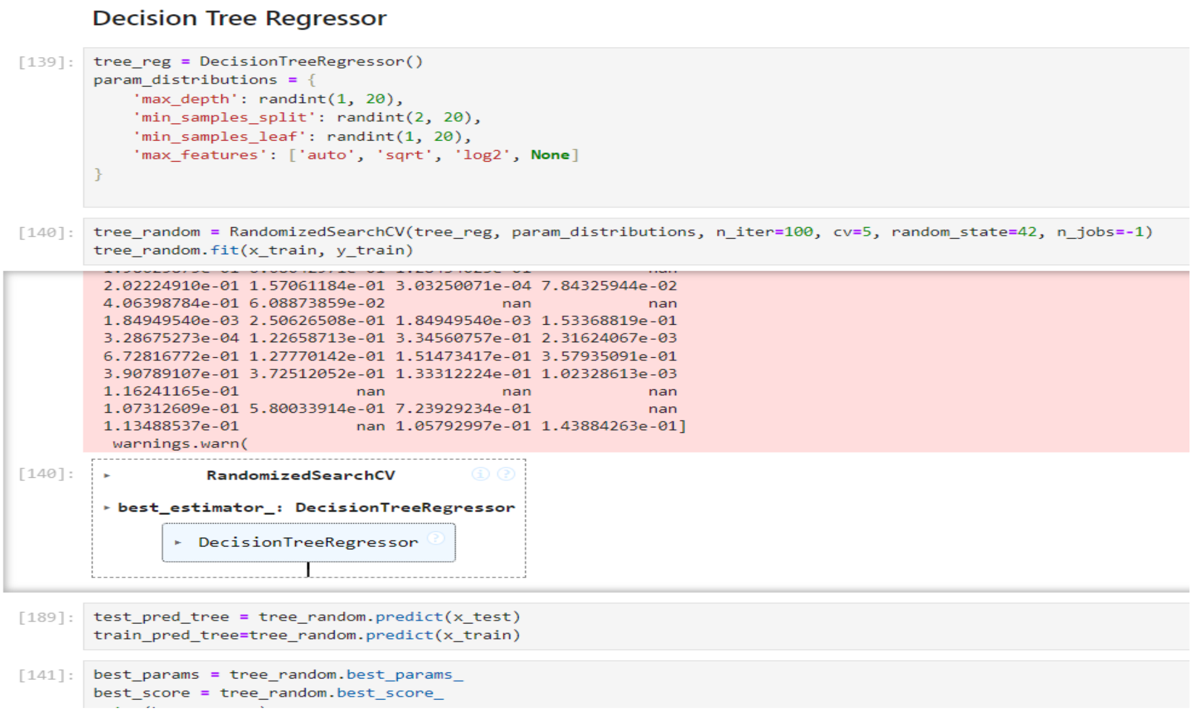
```
[37]: ridgebest_alpha = ridge_random.best_params_
      best_score = ridge_random.best_score_
      print(best_score)
      print(ridgebest_alpha)

      0.0002386836157977523
      {'alpha': 6866.488450042998}
```

```
[38]: ridge_random.score(x_test,y_test)

[38]: 0.0002686734707454397
```

```
[61]: y_pred3=ridge_random.predict(x_test)
```

```
[63]: mse5=mean_squared_error(y_test,y_pred3)
      mse5

[63]: 2.3664620741619075e+27
```

3)



Decision Tree Regressor

```
[139]: tree_reg = DecisionTreeRegressor()
       param_distributions = {
           'max_depth': randint(1, 20),
           'min_samples_split': randint(2, 20),
           'min_samples_leaf': randint(1, 20),
           'max_features': ['auto', 'sqrt', 'log2', None]
       }
```

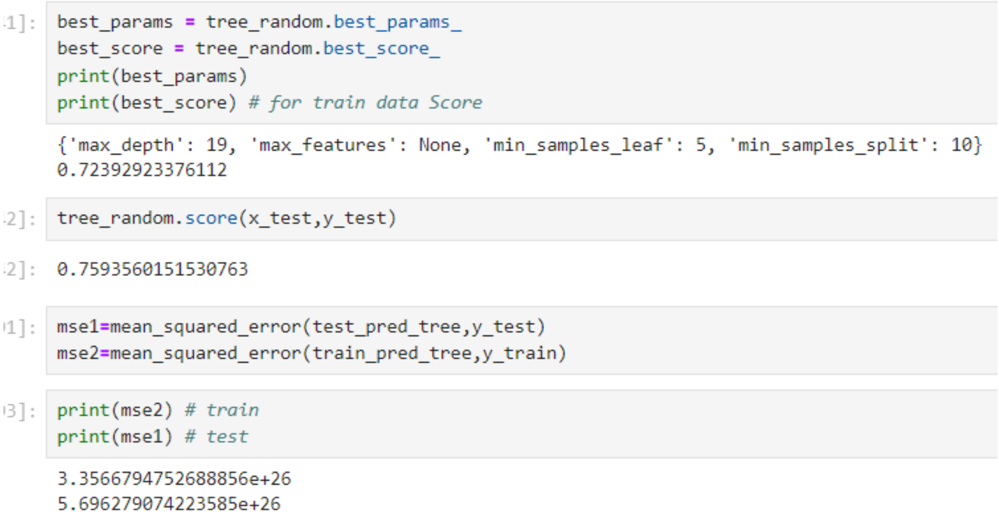
```
[140]: tree_random = RandomizedSearchCV(tree_reg, param_distributions, n_iter=100, cv=5, random_state=42, n_jobs=-1)
       tree_random.fit(x_train, y_train)
```

```
 2.02224910e-01 1.57061184e-01 3.03250071e-04 7.84325944e-02
 4.06398784e-01 6.08873859e-02            nan            nan
 1.84949540e-03 2.50626508e-01 1.84949540e-03 1.53368819e-01
 3.28675273e-04 1.22658713e-01 3.34560757e-01 2.31624067e-03
 6.72816772e-01 1.27770142e-01 1.51473417e-01 3.57935091e-01
 3.90789107e-01 3.72512052e-01 1.33312224e-01 1.02328613e-03
 1.16241165e-01            nan            nan            nan
 1.07312609e-01 5.80033914e-01 7.23929234e-01            nan
 1.13488537e-01            nan 1.05792997e-01 1.43884263e-01]
  warnings.warn(
```

```
[140]: ▸         RandomizedSearchCV
       ▸ best_estimator_: DecisionTreeRegressor
             ▸ DecisionTreeRegressor
```

```
[189]: test_pred_tree = tree_random.predict(x_test)
       train_pred_tree=tree_random.predict(x_train)
```

```
[141]: best_params = tree_random.best_params_
       best_score = tree_random.best_score_
```



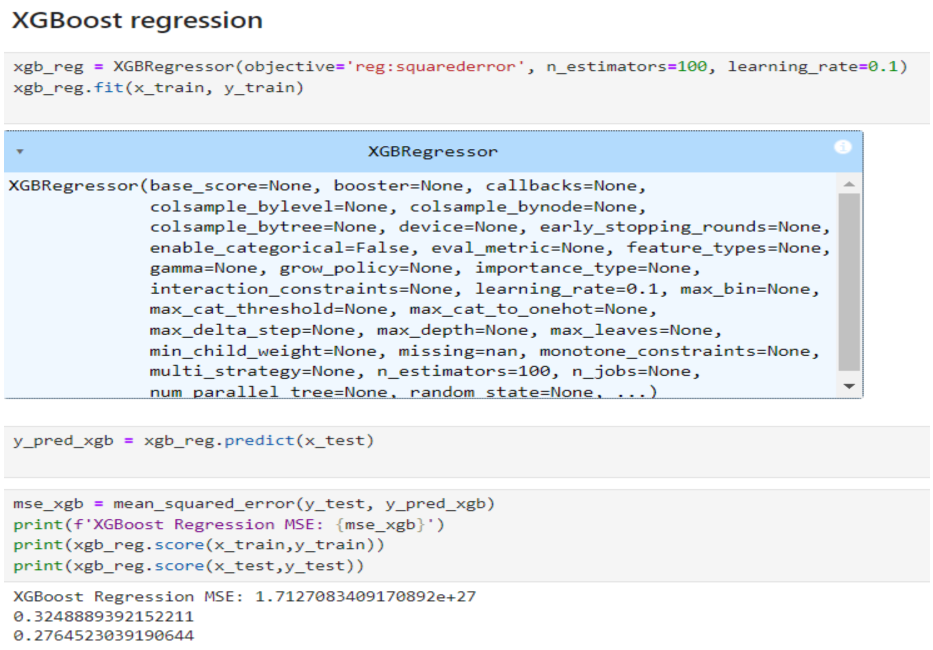
```
1]: best_params = tree_random.best_params_
    best_score = tree_random.best_score_
    print(best_params)
    print(best_score) # for train data Score

    {'max_depth': 19, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 10}
    0.72392923376112
```

```
2]: tree_random.score(x_test,y_test)

2]: 0.7593560151530763
```

```
1]: mse1=mean_squared_error(test_pred_tree,y_test)
    mse2=mean_squared_error(train_pred_tree,y_train)
```

```
3]: print(mse2) # train
    print(mse1) # test

    3.3566794752688856e+26
    5.696279074223585e+26
```

4)



XGBoost regression

```
xgb_reg = XGBRegressor(objective='reg:squarederror', n_estimators=100, learning_rate=0.1)
xgb_reg.fit(x_train, y_train)
```

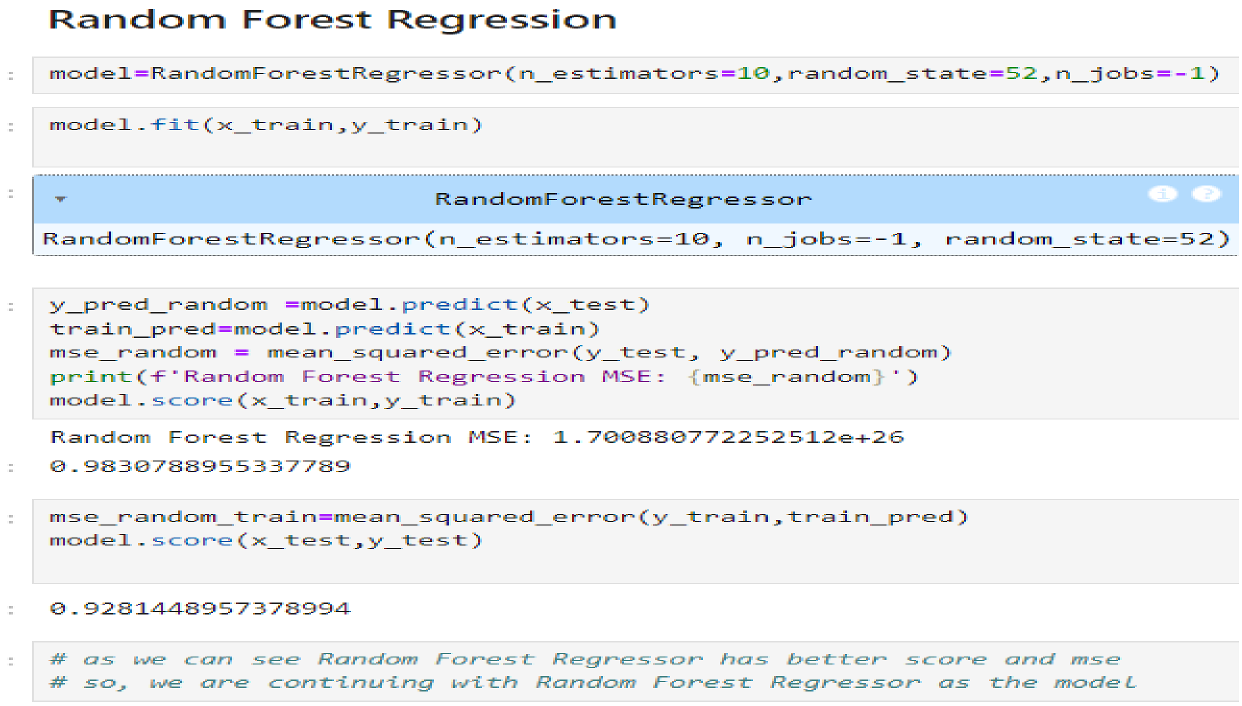
```
XGBRegressor

XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=0.1, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=100, n_jobs=None,
             num_parallel_tree=None, random_state=None, ...)
```

```
y_pred_xgb = xgb_reg.predict(x_test)
```

```
mse_xgb = mean_squared_error(y_test, y_pred_xgb)
print(f'XGBoost Regression MSE: {mse_xgb}')
print(xgb_reg.score(x_train,y_train))
print(xgb_reg.score(x_test,y_test))
```

```
XGBoost Regression MSE: 1.7127083409170892e+27
0.3248889392152211
0.2764523039190644
```

5)



Random Forest Regression

```
model=RandomForestRegressor(n_estimators=10,random_state=52,n_jobs=-1)
```

```
model.fit(x_train,y_train)
```

```
RandomForestRegressor

RandomForestRegressor(n_estimators=10, n_jobs=-1, random_state=52)
```

```
y_pred_random =model.predict(x_test)
train_pred=model.predict(x_train)
mse_random = mean_squared_error(y_test, y_pred_random)
print(f'Random Forest Regression MSE: {mse_random}')
model.score(x_train,y_train)
```

```
Random Forest Regression MSE: 1.700880772252512e+26
0.9830788955337789
```

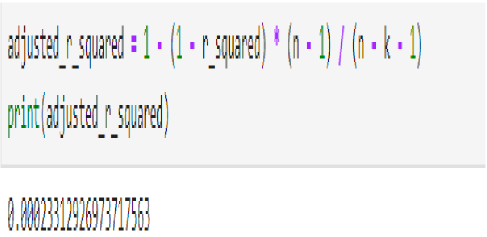
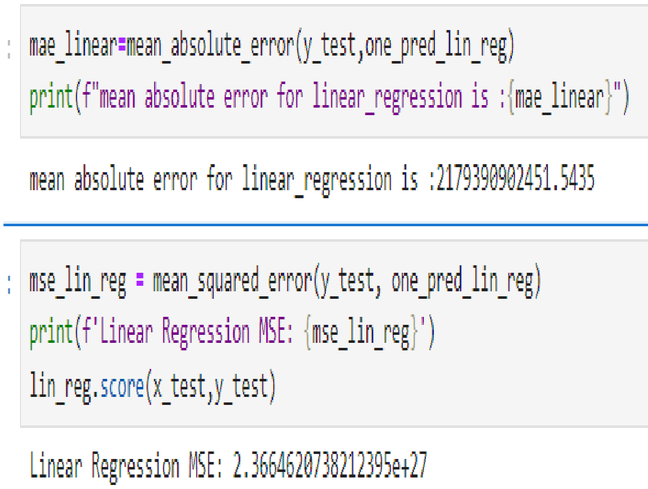
```
mse_random_train=mean_squared_error(y_train,train_pred)
model.score(x_test,y_test)
```

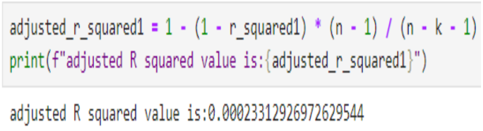
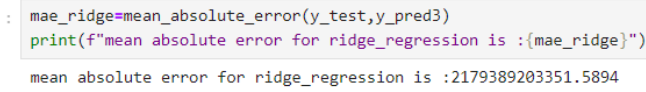
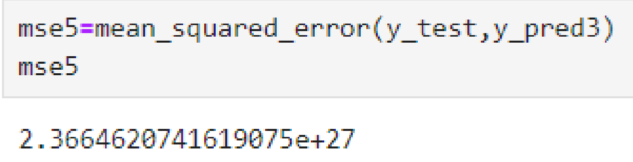
```
0.9281448957378994
```

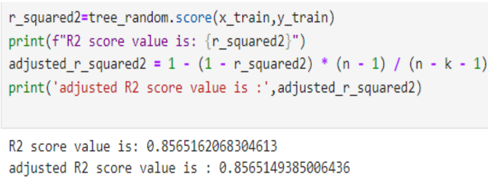
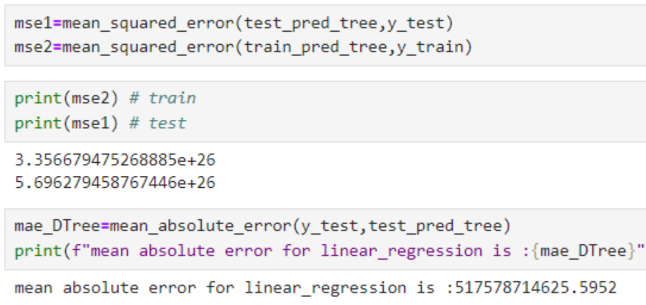
```
# as we can see Random Forest Regressor has better score and mse
# so, we are continuing with Random Forest Regressor as the model
```

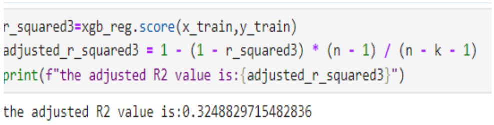
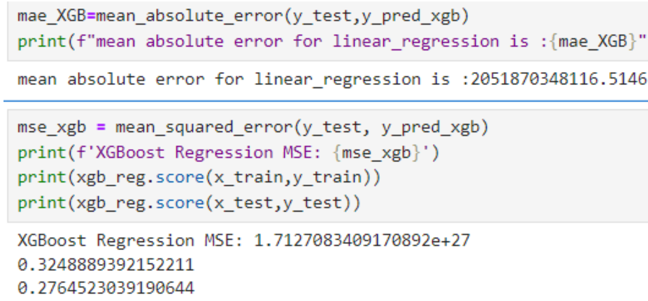
Model Validation and Evaluation Report:

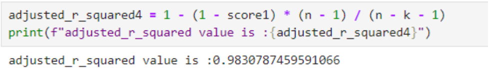
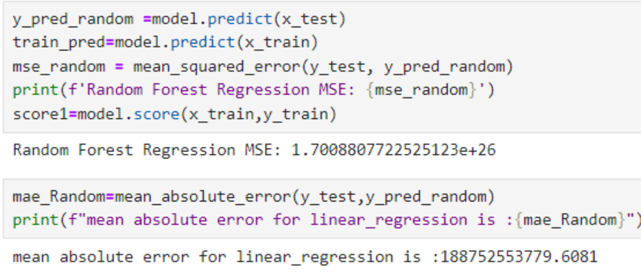
Model	Classification Report (Adjusted R2 Score)	R2 score	Mean Squared Error (MSE) And Mean Absolute Error

Linear Regression	
	`adjusted_r_squared = 1 - (1 - r_squared) * (n - 1) / (n - k - 1)`
	`print(adjusted_r_squared)`
	`0.00023312926973717563`
		0.00024	
			`mae_linear=mean_absolute_error(y_test,one_pred_lin_reg)`
			`print(f"mean absolute error for linear_regression is :{mae_linear}")`
			`mean absolute error for linear_regression is :2179390902451.5435`
			`mse_lin_reg = mean_squared_error(y_test, one_pred_lin_reg)`
			`print(f'Linear Regression MSE: {mse_lin_reg}')`
			`lin_reg.score(x_test,y_test)`
			`Linear Regression MSE: 2.3664620738212395e+27`

Ridge Regression	
	`adjusted_r_squared1 = 1 - (1 - r_squared1) * (n - 1) / (n - k - 1)`
	`print(f"adjusted R squared value is:{adjusted_r_squared1}")`
	`adjusted R squared value is:0.00023312926972629544`
		0.00026	
			`mae_ridge=mean_absolute_error(y_test,y_pred3)`
			`print(f"mean absolute error for ridge_regression is :{mae_ridge}")`
			`mean absolute error for ridge_regression is :2179389203351.5894`
			
			`mse5=mean_squared_error(y_test,y_pred3)`
			`mse5`
			`2.3664620741619075e+27`

Decision Tree Regressor	
	`r_squared2=tree_random.score(x_train,y_train)`
	`print(f"R2 score value is: {r_squared2}")`
	`adjusted_r_squared2 = 1 - (1 - r_squared2) * (n - 1) / (n - k - 1)`
	`print('adjusted R2 score value is :',adjusted_r_squared2)`
	`R2 score value is: 0.8565162068304613`
	`adjusted R2 score value is : 0.8565149385006436`
		0.75935	
			`mse1=mean_squared_error(test_pred_tree,y_test)`
			`mse2=mean_squared_error(train_pred_tree,y_train)`
			`print(mse2) # train`
			`print(mse1) # test`
			`3.356679475268885e+26`
			`5.696279458767446e+26`
			`mae_DTree=mean_absolute_error(y_test,test_pred_tree)`
			`print(f"mean absolute error for linear_regression is :{mae_DTree}"`
			`mean absolute error for linear_regression is :517578714625.5952`

XG boost Regression	
	`r_squared3=xgb_reg.score(x_train,y_train)`
	`adjusted_r_squared3 = 1 - (1 - r_squared3) * (n - 1) / (n - k - 1)`
	`print(f"the adjusted R2 value is:{adjusted_r_squared3}")`
	`the adjusted R2 value is:0.3248829715482836`
		0.32488	
			`mae_XGB=mean_absolute_error(y_test,y_pred_xgb)`
			`print(f"mean absolute error for linear_regression is :{mae_XGB}"`
			`mean absolute error for linear_regression is :2051870348116.5146`
			`mse_xgb = mean_squared_error(y_test, y_pred_xgb)`
			`print(f'XGBoost Regression MSE: {mse_xgb}')`
			`print(xgb_reg.score(x_train,y_train))`
			`print(xgb_reg.score(x_test,y_test))`
			`XGBoost Regression MSE: 1.7127083409170892e+27`
			`0.3248889392152211`
			`0.2764523039190644`

Random Forest Regressor	
	`adjusted_r_squared4 = 1 - (1 - score1) * (n - 1) / (n - k - 1)`
	`print(f"adjusted_r_squared value is :{adjusted_r_squared4}")`
	`adjusted_r_squared value is :0.9830787459591066`
		0.98307	
			`y_pred_random =model.predict(x_test)`
			`train_pred=model.predict(x_train)`
			`mse_random = mean_squared_error(y_test, y_pred_random)`
			`print(f'Random Forest Regression MSE: {mse_random}')`
			`score1=model.score(x_train,y_train)`
			`Random Forest Regression MSE: 1.7008807722525123e+26`
			`mae_Random=mean_absolute_error(y_test,y_pred_random)`
			`print(f"mean absolute error for linear_regression is :{mae_Random}")`
			`mean absolute error for linear_regression is :188752553779.6081`