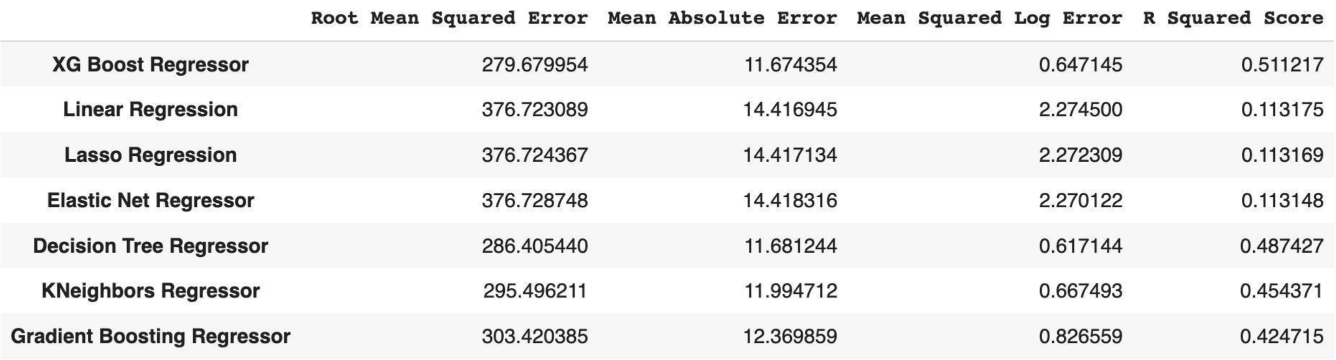
Project Development Phase
Model Performance Test

Date	19 November 2022
Team ID	**PNT2022TMID15280**
Project Name	Project - **DemandEst - AI powered Food Demand Forecaster**
Maximum Marks	10 Marks

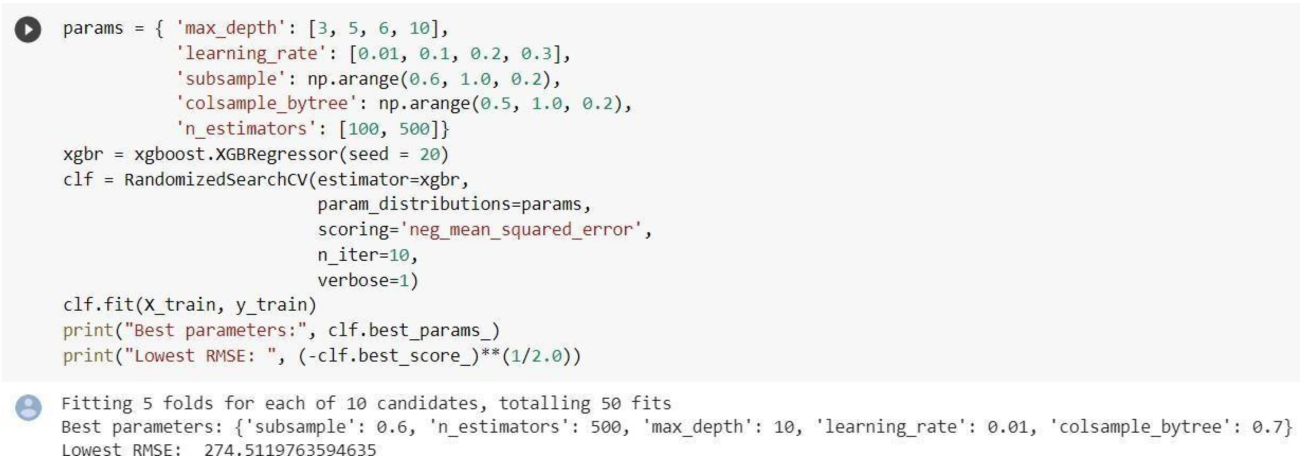
Model Performance Testing:

Project team shall fill the following information in the model performance testing template.

S.No.	Parameter	Values	Screenshot
1.	Metrics	**XGBRegressor Model:** *MAE* - 11.674354, *MSE* - 0.647145 *RMSE* - 279.679954, *R2 score* - 0.511217	attached below
2.	Tune the Model	**Hyperparameter Tuning -** RandomizedSearchCV, GridSearchCV **Validation Method -** KFold Cross Validation	attached below

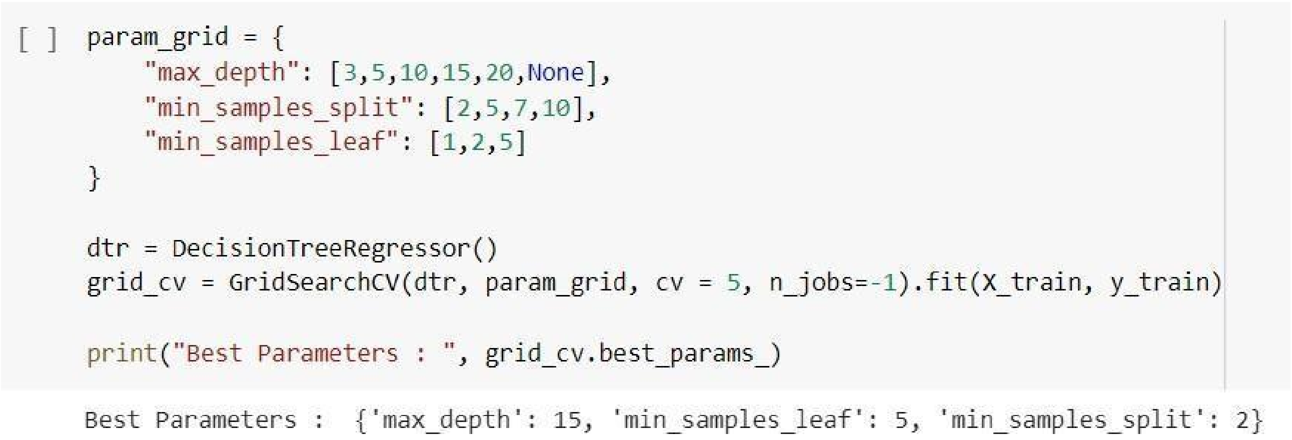


	Root Mean Squared Error	Mean Absolute Error	Mean Squared Log Error	R Squared Score
XG Boost Regressor	279.679954	11.674354	0.647145	0.511217
Linear Regression	376.723089	14.416945	2.274500	0.113175
Lasso Regression	376.724367	14.417134	2.272309	0.113169
Elastic Net Regressor	376.728748	14.418316	2.270122	0.113148
Decision Tree Regressor	286.405440	11.681244	0.617144	0.487427
KNeighbors Regressor	295.496211	11.994712	0.667493	0.454371
Gradient Boosting Regressor	303.420385	12.369859	0.826559	0.424715



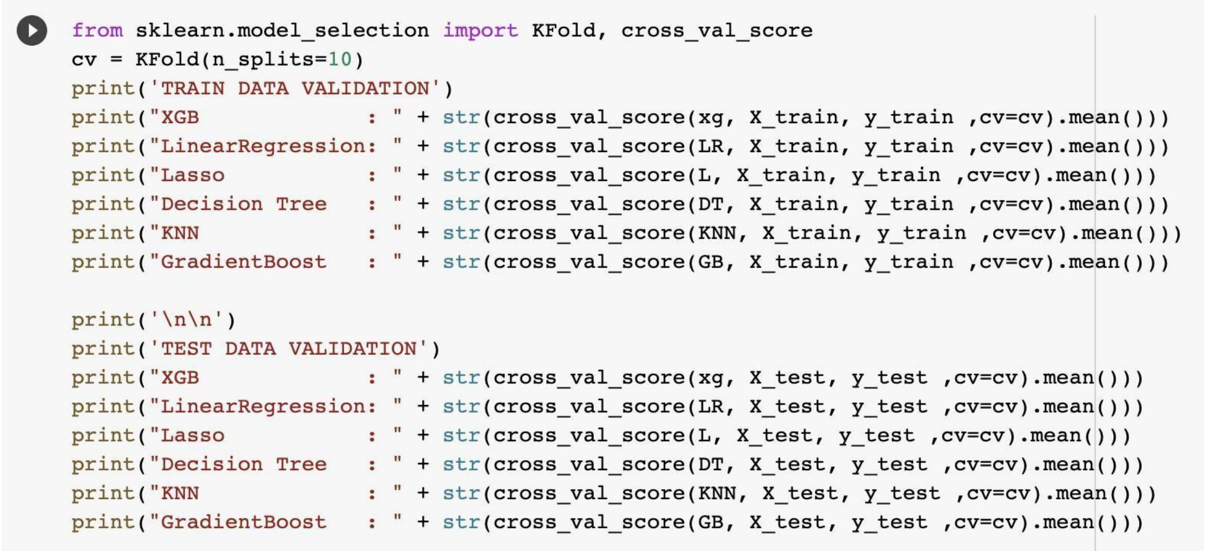
```
params = { 'max_depth': [3, 5, 6, 10],
           'learning_rate': [0.01, 0.1, 0.2, 0.3],
           'subsample': np.arange(0.6, 1.0, 0.2),
           'colsample_bytree': np.arange(0.5, 1.0, 0.2),
           'n_estimators': [100, 500]}
xgbr = xgboost.XGBRegressor(seed = 20)
clf = RandomizedSearchCV(estimator=xgbr,
                         param_distributions=params,
                         scoring='neg_mean_squared_error',
                         n_iter=10,
                         verbose=1)
clf.fit(X_train, y_train)
print("Best parameters:", clf.best_params_)
print("Lowest RMSE: ", (-clf.best_score_)**(1/2.0))
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
Best parameters: {'subsample': 0.6, 'n_estimators': 500, 'max_depth': 10, 'learning_rate': 0.01, 'colsample_bytree': 0.7}
Lowest RMSE:  274.5119763594635
```

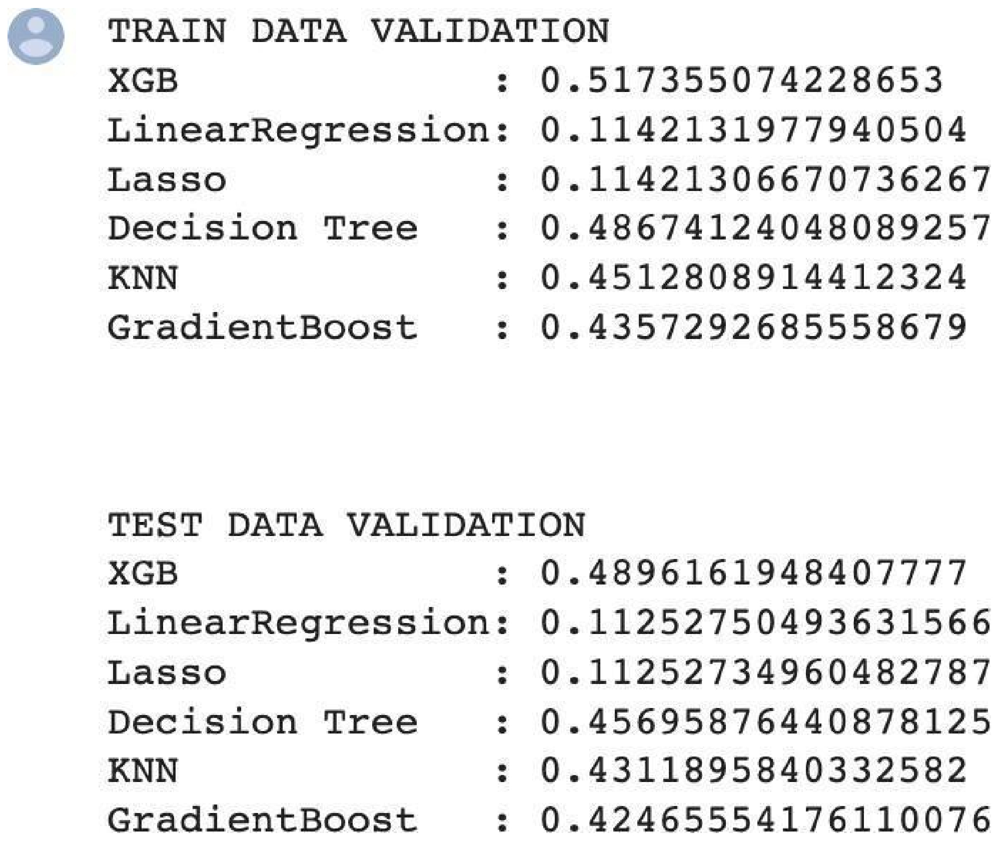


```
param_grid = {
    "max_depth": [3,5,10,15,20,None],
    "min_samples_split": [2,5,7,10],
    "min_samples_leaf": [1,2,5]
}

dtr = DecisionTreeRegressor()
grid_cv = GridSearchCV(dtr, param_grid, cv = 5, n_jobs=-1).fit(X_train, y_train)

print("Best Parameters : ", grid_cv.best_params_)
```

```
Best Parameters :  {'max_depth': 15, 'min_samples_leaf': 5, 'min_samples_split': 2}
```



```
from sklearn.model_selection import KFold, cross_val_score
cv = KFold(n_splits=10)
print('TRAIN DATA VALIDATION')
print("XGB              : " + str(cross_val_score(xg, X_train, y_train ,cv=cv).mean()))
print("LinearRegression: " + str(cross_val_score(LR, X_train, y_train ,cv=cv).mean()))
print("Lasso            : " + str(cross_val_score(L, X_train, y_train ,cv=cv).mean()))
print("Decision Tree    : " + str(cross_val_score(DT, X_train, y_train ,cv=cv).mean()))
print("KNN              : " + str(cross_val_score(KNN, X_train, y_train ,cv=cv).mean()))
print("GradientBoost    : " + str(cross_val_score(GB, X_train, y_train ,cv=cv).mean()))

print('\n\n')
print('TEST DATA VALIDATION')
print("XGB              : " + str(cross_val_score(xg, X_test, y_test ,cv=cv).mean()))
print("LinearRegression: " + str(cross_val_score(LR, X_test, y_test ,cv=cv).mean()))
print("Lasso            : " + str(cross_val_score(L, X_test, y_test ,cv=cv).mean()))
print("Decision Tree    : " + str(cross_val_score(DT, X_test, y_test ,cv=cv).mean()))
print("KNN              : " + str(cross_val_score(KNN, X_test, y_test ,cv=cv).mean()))
print("GradientBoost    : " + str(cross_val_score(GB, X_test, y_test ,cv=cv).mean()))
```



```
TRAIN DATA VALIDATION
XGB              : 0.517355074228653
LinearRegression: 0.1142131977940504
Lasso            : 0.11421306670736267
Decision Tree    : 0.48674124048089257
KNN              : 0.4512808914412324
GradientBoost    : 0.4357292685558679



TEST DATA VALIDATION
XGB              : 0.4896161948407777
LinearRegression: 0.11252750493631566
Lasso            : 0.11252734960482787
Decision Tree    : 0.45695876440878125
KNN              : 0.4311895840332582
GradientBoost    : 0.42465554176110076
```