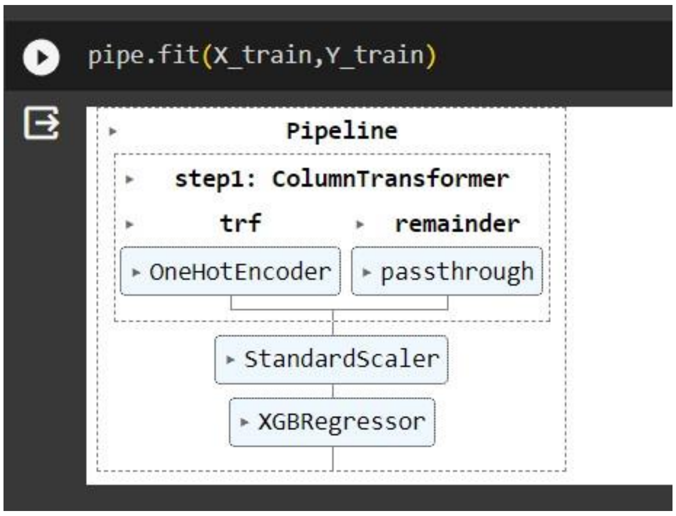
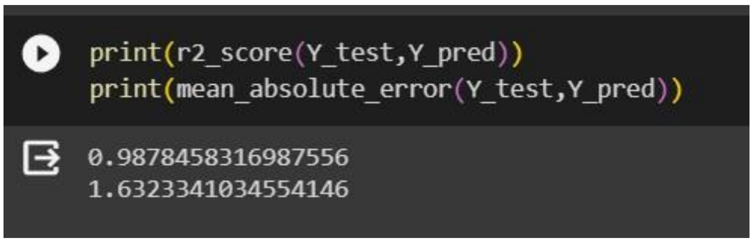
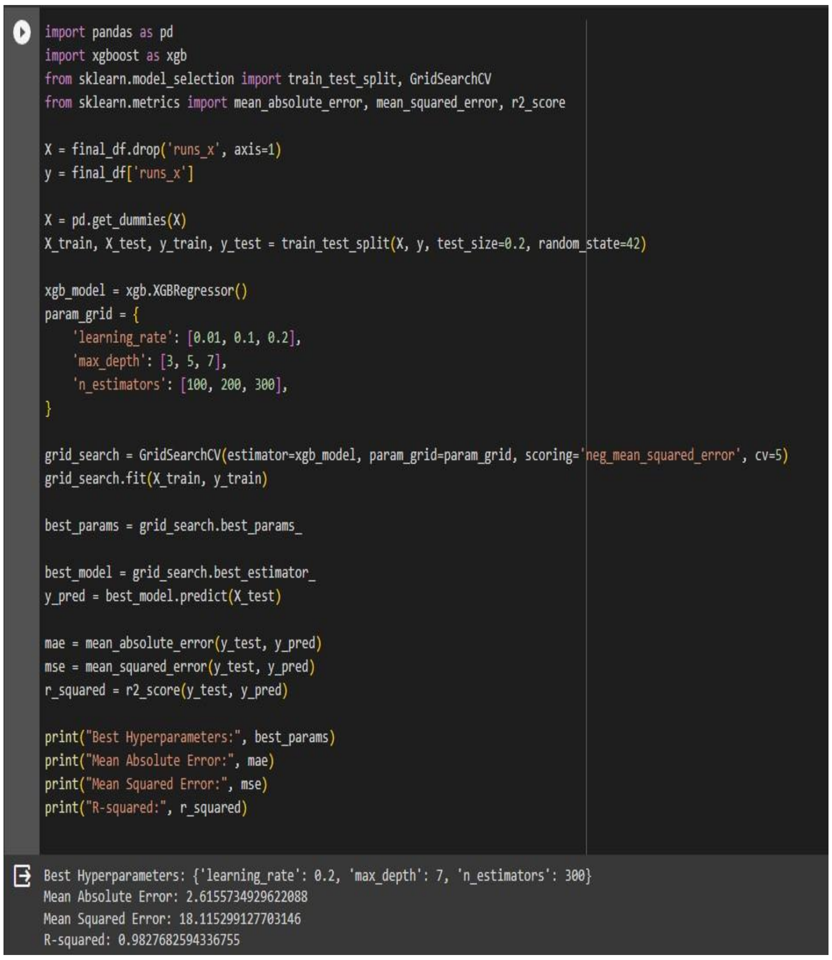
Project Development Phase
(Model Performance Test)

Date	20-11-2023
Team ID	Team-592914
Project Name	T20 Totalitarian: Mastering Score Predictions
Maximum Marks	10 Marks

S.No.	Parameter	Values	Screenshot

1.	Metrics	**Regression Model:** Utilizing preprocessed data and the chosen features, the XGBoost model has been implemented. XGBoost is an advanced gradient boosting algorithm employing decision trees as its fundamental learners. **Classification Model:** Assess the effectiveness of the XGBoost model by gauging its performance using metrics like mean absolute error, mean squared error, and R-squared.	 `pipe.fit(X_train,Y_train)` Pipeline step1: ColumnTransformer trf remainder OneHotEncoder passthrough StandardScaler XGBRegressor  `print(r2_score(Y_test,Y_pred))` `print(mean_absolute_error(Y_test,Y_pred))` `0.9878458316987556` `1.6323341034554146`

2.	Tune the Model	Hyperparameter Tuning - Validation Method -	 In conclusion, the hyperparameter tuning and validation process for the XGBoost model on the 'final_df' dataset resulted in the identification of optimal parameters. The model exhibits lower Mean Absolute Error (MAE) and Mean Squared Error (MSE) values, indicating minimal prediction errors. Moreover, the R-squared value, approaching 1, signifies a strong alignment between the model and the dataset. This suggests that the chosen XGBoost model, with the fine-tuned hyperparameters, is robust and performs exceptionally well, making it an excellent fit for the project's objectives.

```python
import pandas as pd
import xgboost as xgb
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

X = final_df.drop('runs_x', axis=1)
y = final_df['runs_x']

X = pd.get_dummies(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

xgb_model = xgb.XGBRegressor()
param_grid = {
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 5, 7],
    'n_estimators': [100, 200, 300],
}

grid_search = GridSearchCV(estimator=xgb_model, param_grid=param_grid, scoring='neg_mean_squared_error', cv=5)
grid_search.fit(X_train, y_train)

best_params = grid_search.best_params_

best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r_squared = r2_score(y_test, y_pred)

print("Best Hyperparameters:", best_params)
print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("R-squared:", r_squared)
```

```
Best Hyperparameters: {'learning_rate': 0.2, 'max_depth': 7, 'n_estimators': 300}
Mean Absolute Error: 2.6155734929622088
Mean Squared Error: 18.115299127703146
R-squared: 0.9827682594336755
```