

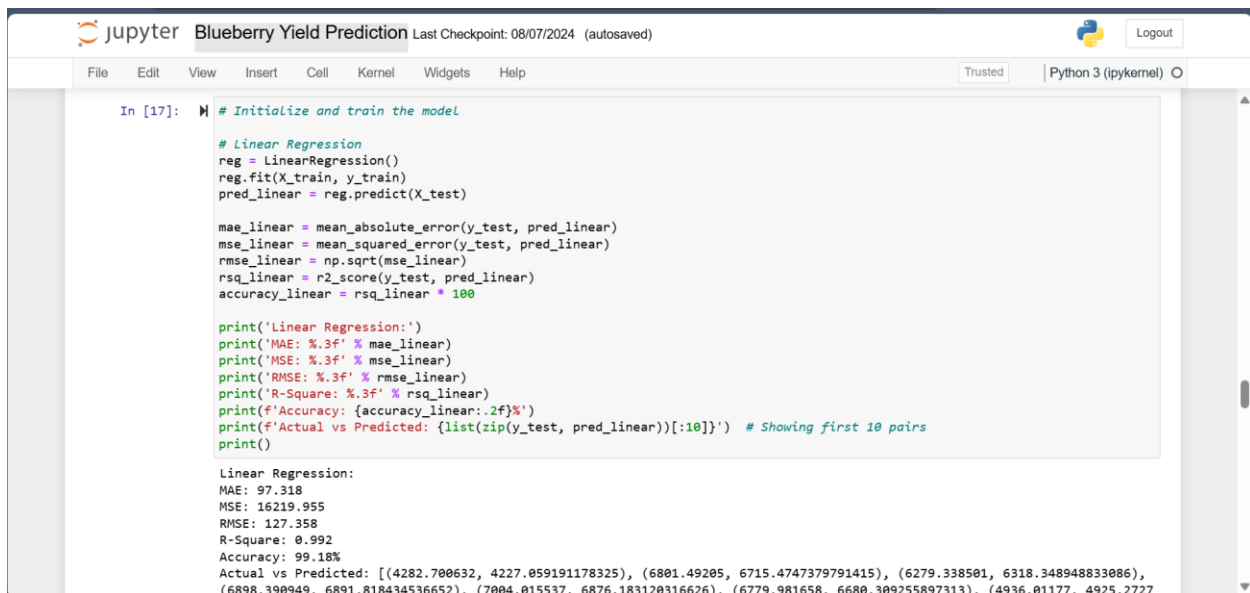
Model Development Phase Template

Date	20 July 2024
Team ID	SWTID1721319573
Project Title	Blueberry Yield Prediction
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:



The screenshot shows a Jupyter Notebook titled "Blueberry Yield Prediction" with the following code and output:

```
In [17]: # Initialize and train the model

# Linear Regression
reg = LinearRegression()
reg.fit(X_train, y_train)
pred_linear = reg.predict(X_test)

mae_linear = mean_absolute_error(y_test, pred_linear)
mse_linear = mean_squared_error(y_test, pred_linear)
rmse_linear = np.sqrt(mse_linear)
rsq_linear = r2_score(y_test, pred_linear)
accuracy_linear = rsq_linear * 100

print('Linear Regression:')
print('MAE: %.3f' % mae_linear)
print('MSE: %.3f' % mse_linear)
print('RMSE: %.3f' % rmse_linear)
print('R-Square: %.3f' % rsq_linear)
print(f'Accuracy: {accuracy_linear:.2f}%')
print(f'Actual vs Predicted: {list(zip(y_test, pred_linear))[:10]}') # Showing first 10 pairs
print()
```

Linear Regression:
 MAE: 97.318
 MSE: 16219.955
 RMSE: 127.358
 R-Square: 0.992
 Accuracy: 99.18%
 Actual vs Predicted: [(4282.700632, 4227.059191178325), (6801.49205, 6715.4747379791415), (6279.338501, 6318.348948833086), (6898.390949, 6891.818434536652), (7004.015537, 6876.183120316626), (6779.981658, 6680.309255897313), (4936.01177, 4925.2727)

```
57159071), (7604.315838, 7630.315686537314), (3723.523376, 3894.539832186767), (6037.686131, 6029.792189965131)]

In [18]: # Random Forest Regressor
rf_reg = RandomForestRegressor()
rf_reg.fit(X_train, y_train)
pred_rf = rf_reg.predict(X_test)

mae_rf = mean_absolute_error(y_test, pred_rf)
mse_rf = mean_squared_error(y_test, pred_rf)
rmse_rf = np.sqrt(mse_rf)
rsq_rf = r2_score(y_test, pred_rf)
accuracy_rf = rsq_rf * 100

print('Random Forest Regressor:')
print('MAE: %.3f' % mae_rf)
print('MSE: %.3f' % mse_rf)
print('RMSE: %.3f' % rmse_rf)
print('R-Square: %.3f' % rsq_rf)
print('Accuracy: {accuracy_rf:.2f}%')
print('Actual vs Predicted: {list(zip(y_test, pred_rf))[:10]}') # Showing first 10 pairs
print()

Random Forest Regressor:
MAE: 116.868
MSE: 22604.657
RMSE: 150.348
R-Square: 0.989
Accuracy: 98.86%
Actual vs Predicted: [(4282.700632, 4139.818938290001), (6801.49205, 6682.12248011), (6279.338501, 6233.448528890003), (689
8.390949, 6904.70566688), (7004.015537, 6904.661148249991), (6779.981658, 6616.757262779998), (4936.01177, 4787.94731337000
3), (7604.315838, 7584.481504839996), (3723.523376, 4009.5987002700003), (6037.686131, 6006.676942220003)]
```

```
In [19]: # Decision Tree Regressor
dt_reg = DecisionTreeRegressor()
dt_reg.fit(X_train, y_train)
pred_dt = dt_reg.predict(X_test)

mae_dt = mean_absolute_error(y_test, pred_dt)
mse_dt = mean_squared_error(y_test, pred_dt)
rmse_dt = np.sqrt(mse_dt)
rsq_dt = r2_score(y_test, pred_dt)
accuracy_dt = rsq_dt * 100

print('Decision Tree Regressor:')
print('MAE: %.3f' % mae_dt)
print('MSE: %.3f' % mse_dt)
print('RMSE: %.3f' % rmse_dt)
print('R-Square: %.3f' % rsq_dt)
print('Accuracy: {accuracy_dt:.2f}%')
print('Actual vs Predicted: {list(zip(y_test, pred_dt))[:10]}') # Showing first 10 pairs
print()

Decision Tree Regressor:
MAE: 155.595
MSE: 42571.611
RMSE: 206.329
R-Square: 0.978
Accuracy: 97.84%
Actual vs Predicted: [(4282.700632, 4125.757119), (6801.49205, 6619.846953), (6279.338501, 6397.355689), (6898.390949, 6922.
846792), (7004.015537, 6575.592668), (6779.981658, 6771.722906), (4936.01177, 4945.794431), (7604.315838, 7576.39253), (372
3.523376, 4178.772056), (6037.686131, 6107.382466)]
```

```
In [21]: # XGBoost Regressor
xgb_reg = XGBRegressor()
xgb_reg.fit(X_train, y_train)
pred_xgb = xgb_reg.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)
accuracy_xgb = rsq_xgb * 100

print('XGBoost Regressor:')
print('MAE: %.3f' % mae_xgb)
print('MSE: %.3f' % mse_xgb)
print('RMSE: %.3f' % rmse_xgb)
print('R-Square: %.3f' % rsq_xgb)
print('Accuracy: {accuracy_xgb:.2f}%')
print('Actual vs Predicted: {list(zip(y_test, pred_xgb))[:10]}') # Showing first 10 pairs
print()

XGBoost Regressor:
MAE: 111.927
MSE: 20632.639
RMSE: 143.641
R-Square: 0.990
```

```

Accuracy: 98.96%
Actual vs Predicted: [(4282.700632, 4242.088), (6801.49205, 6684.846), (6279.338501, 6362.355), (6898.390949, 6913.2456), (7004.015537, 6884.957), (6779.981658, 6698.55), (4936.01177, 4913.695), (7604.315838, 7600.5967), (3723.523376, 3954.2666), (6037.686131, 6095.0405)]

In [22]: # Find the best model
r2_scores = {
    "Linear Regression": rsq_linear,
    "Random Forest": rsq_rf,
    "Decision Tree": rsq_dt, # Make sure the variable name matches
    "XGBoost": rsq_xgb
}

best_model_name = max(r2_scores, key=r2_scores.get)
print(f"\nBest Model: {best_model_name}")

# Assign the best model to best_model variable
if best_model_name == 'Linear Regression':
    best_model = reg
    y_pred = pred_linear
elif best_model_name == 'Random Forest':
    best_model = rf_reg # Corrected variable name
    y_pred = pred_rf
elif best_model_name == 'Decision Tree':
    best_model = dt_reg # Corrected variable name
    y_pred = pred_dt
elif best_model_name == 'XGBoost':
    best_model = xgb_reg # Corrected variable name
    y_pred = pred_xgb

# Assign the best model to best_model variable
if best_model_name == 'Linear Regression':
    best_model = reg
    y_pred = pred_linear
elif best_model_name == 'Random Forest':
    best_model = rf_reg # Corrected variable name
    y_pred = pred_rf
elif best_model_name == 'Decision Tree':
    best_model = dt_reg # Corrected variable name
    y_pred = pred_dt
elif best_model_name == 'XGBoost':
    best_model = xgb_reg # Corrected variable name
    y_pred = pred_xgb

Best Model: Linear Regression

```

Model Validation and Evaluation Report:

Model	Regression Performance Summary	Prediction Accuracy	Residual Analysis
Linear Regression	MAE: 97.318 MSE: 16219.955 RMSE: 127.358 R-Square: 0.992	Accuracy: 99.18%	Actual vs Predicted: [(4282.700632, 4227.059191178325), (6801.49205, 6715.4747379791415), (6279.338501, 6318.348948833086), (6898.390949, 6891.818434536652), (7004.015537, 6876.183120316626), (6779.981658, 6680.309255897313),

			(4936.01177, 4925.272757159071), (7604.315838, 7630.315686537314), (3723.523376, 3894.539832186767), (6037.686131, 6029.792189965131)]
Random Forest Regressor	MAE: 117.197 MSE: 22845.764 RMSE: 151.148 R-Square: 0.988	Accuracy: 98.86%	Actual vs Predicted: [(4282.700632, 4139.818938290001), (6801.49205, 6682.12248011), (6279.338501, 6233.448528890003), (6898.390949, 6904.70566688), (7004.015537, 6904.661148249991), (6779.981658, 6616.757262779998), (4936.01177, 4787.947313370003), (7604.315838, 7584.481504839996), (3723.523376, 4009.5987002700003), (6037.686131, 6006.676942220003)]
Decision Tree Regressor	MAE: 161.110 MSE: 45330.850 RMSE: 212.910 R-Square: 0.977	Accuracy: 97.84%	Actual vs Predicted: [(4282.700632, 4125.757119), (6801.49205, 6619.846953), (6279.338501, 6397.355689),

			(6898.390949, 6922.846792), (7004.015537, 6575.592668), (6779.981658, 6771.722906), (4936.01177, 4945.794431), (7604.315838, 7576.39253), (3723.523376, 4178.772056), (6037.686131, 6107.382466)]
XGBoost Regressor	MAE: 111.927 MSE: 20632.639 RMSE: 143.641 R-Square: 0.990	Accuracy: 98.96%	Actual vs Predicted: [(4282.700632, 4242.088), (6801.49205, 6684.846), (6279.338501, 6362.355), (6898.390949, 6913.2456), (7004.015537, 6884.957), (6779.981658, 6698.55), (4936.01177, 4913.695), (7604.315838, 7600.5967), (3723.523376, 3954.2666), (6037.686131, 6095.0405)]