feature_engineering

April 26, 2024

[ ]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier,␣
 ↪GradientBoostingRegressor
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
```

[ ]:
```
data = pd.read_csv('./final_data/final_data.csv')
```

[ ]:
```
# Feature engineering using polynomial features
poly_features = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly_features.fit_transform(data[['stint length', 'avg_lap_time']])
poly_feature_names = [f'x{i}' for i in range(1, X_poly.shape[1] + 1)]
data_poly = pd.DataFrame(X_poly, columns=poly_feature_names)
data = pd.concat([data, data_poly], axis=1)
```

[ ]:
```
X = data.drop(columns=['compound'])
y_compound = data['compound']
y_stint = data['stint length']
```

[ ]:
```
X_encoded = pd.get_dummies(X)
```

[ ]:
```
X_train_compound, X_test_compound, y_train_compound, y_test_compound =␣
 ↪train_test_split(X_encoded, y_compound, test_size=0.2, random_state=42)
```

[ ]:
```
scaler_compound = StandardScaler()
X_train_scaled_compound = scaler_compound.fit_transform(X_train_compound)
X_test_scaled_compound = scaler_compound.transform(X_test_compound)
```

[ ]:
```
# Train Gradient Boosting Classifier for tire compounds prediction
gb_model_compound = GradientBoostingClassifier(n_estimators=100,␣
 ↪random_state=42)
gb_model_compound.fit(X_train_scaled_compound, y_train_compound)
```

[ ]:
```
GradientBoostingClassifier(random_state=42)
```

[ ]:
```
X_train_stint, X_test_stint, y_train_stint, y_test_stint =␣
 ↪train_test_split(X_encoded, y_stint, test_size=0.2, random_state=42)
```

```
[ ]: scaler_stint = StandardScaler()
     X_train_scaled_stint = scaler_stint.fit_transform(X_train_stint)
     X_test_scaled_stint = scaler_stint.transform(X_test_stint)
```

```
[ ]: # Train Gradient Boosting Regressor for stint lengths prediction
     gb_model_stint = GradientBoostingRegressor(n_estimators=100, random_state=42)
     gb_model_stint.fit(X_train_scaled_stint, y_train_stint)
```

```
[ ]: GradientBoostingRegressor(random_state=42)
```

```
[ ]: def predict_tires_and_stints(driver_id, circuit_id, num_stints):
         # Filter data for the given driver_id and circuit_id
         driver_data = data[(data['driverid'] == driver_id) & (data['circuitid'] ==␣
      ↪circuit_id)]

         if driver_data.empty:
             print('No data available for prediction.')
             return [], []

         # Sort the driver data by stint
         sorted_driver_data = driver_data.sort_values(by='stint')

         # Select the specified number of stints
         selected_stints = sorted_driver_data.iloc[:num_stints]

         if selected_stints.empty:
             print(f'No data available for the first {num_stints} stints.')
             return [], []

         # Prepare the input features for prediction
         X_input = selected_stints.drop(columns=['compound'])

         # One-hot encode the input features
         X_input_encoded = pd.get_dummies(X_input)

         if X_input_encoded.empty:
             print('No data available after encoding.')
             return [], []

         # Scale the input features for tire compounds prediction
         X_input_scaled_compound = scaler_compound.transform(X_input_encoded)

         # Predict tire compounds
         predicted_tires = gb_model_compound.predict(X_input_scaled_compound)

         # Scale the input features for stint lengths prediction
         X_input_scaled_stint = scaler_stint.transform(X_input_encoded)
```

```
    # Predict stint lengths
    predicted_stint_lengths = gb_model_stint.predict(X_input_scaled_stint)

    # Round the predicted stint lengths to integers
    predicted_stint_lengths_rounded = [int(round(length)) for length in␣
↪predicted_stint_lengths]

    return list(predicted_tires), predicted_stint_lengths_rounded
```

[ ]:
```
predicted_tires, predicted_stint_lengths = predict_tires_and_stints(4, 1, 2)
print(f'Predicted Tires: {predicted_tires}')
print(f'Predicted Stint Lengths: {predicted_stint_lengths}')
```

```
Predicted Tires: ['ULTRASOFT', 'HARD']
Predicted Stint Lengths: [25, 39]
```