Passing-Yards-and-Touchdowns

October 10, 2025

[1]:
```python
import sqlite3
import pandas as pd
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
```

Your current models, as they stand, are trained on historical data of quarterbacks' past performances and are designed to predict outcomes like passing touchdowns and passing yards based on input features from those past games. Here's how these models are useful and how you might leverage them:

0.0.1 1. Performance Evaluation and Validation:

- **Model Accuracy:** You can use the models to evaluate how well they perform by comparing the predicted outcomes with actual results from historical games. This helps in validating the model's accuracy and understanding its strengths and weaknesses.
- **Understanding Predictive Power:** By seeing how well the model predicts past games, you can gauge its predictive power, which is critical before you apply it to future games.

0.0.2 2. Identifying Trends and Patterns:

- **Historical Analysis:** By applying the models to historical data, you can identify trends and patterns in a quarterback's performance. For example, you might notice that the model consistently over- or under-predicts in certain conditions (e.g., away games, against strong defenses).
- **Model Improvement:** These insights can inform improvements in the model, such as adding new features (e.g., opponent strength, weather conditions) to capture these patterns better.

0.0.3 3. Scenario Analysis and What-If Predictions:

- **Simulating Different Conditions:** You can simulate different scenarios using historical data. For example, by altering the input features slightly, you can predict how the quarterback might have performed if conditions were different (e.g., fewer sacks, more completions).
- **Comparison of Past vs. Predicted:** This allows you to compare the model's predictions with actual outcomes to better understand the impact of certain variables on the quarterback's performance.

0.0.4 4. Model Tuning and Feature Engineering:

- **Feature Importance:** Use the models to analyze which features (e.g., completions, attempts, sacks) are most influential in predicting touchdowns or passing yards. This insight can guide feature engineering to improve model performance.
- **Hyperparameter Tuning:** The models provide a base that you can tune further by adjusting hyperparameters or adding/removing features to optimize performance.

0.0.5 5. Baseline for Predicting Future Games:

- **Foundation for Future Predictions:** Although the models are trained on historical data, they serve as a baseline that can be adapted to predict future games. With the right input data (reflecting upcoming game conditions), these models can be used to forecast a quarterback's performance.
- **Model Transferability:** The logic and structure of these models can be transferred to predict future games. By updating the input data to reflect expected game conditions, the models can be repurposed for forward-looking predictions.

0.0.6 How to Transition to Predicting Future Games:

- **Gather Data for Upcoming Games:** Collect or estimate the input features (e.g., expected completions, attempts, interceptions) for the upcoming game based on analysis or expert judgment.
- **Use the Existing Models:** Input this data into your existing models to generate predictions for future games.
- **Compare with Actual Outcomes:** After the game, compare your model's predictions with the actual performance to further refine your model.

0.0.7 Conclusion:

Your current models are valuable tools for understanding past performance, validating the model's effectiveness, and serving as a baseline for future predictions. By transitioning from historical data to forecasted input features, these models can be adapted to predict future outcomes and be a powerful asset in decision-making, whether for betting, fantasy sports, or strategic analysis.

[23]:
```
# XGBoost OG 1 - Predicting QB Performance (TD's)
# For backtesting features

conn = sqlite3.connect('nfl.db')
query = "SELECT * FROM PlayerStats"
player_stats_df = pd.read_sql_query(query, conn)
conn.close()

# Filter the data for quarterbacks only (position == 'QB')
qb_data = player_stats_df[player_stats_df['position'] == 'QB']

# Select relevant features for predicting passing touchdowns
features = ['completions', 'attempts', 'passing_yards', 'interceptions',␣
 ↪'sacks']
```

```
target = 'passing_tds'

# Define X and y
X = qb_data[features]
y = qb_data[target]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Create and train the XGBoost regressor model
model = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=100,␣
 ↪learning_rate=0.1, max_depth=3)
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model's performance
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')

# Feature importance
importance = model.feature_importances_
features_importance_df = pd.DataFrame({'Feature': features, 'Importance':␣
 ↪importance})
print(features_importance_df.sort_values(by='Importance', ascending=False))
```

```
Mean Squared Error: 0.7296015093903238
         Feature  Importance
2  passing_yards    0.719514
3  interceptions    0.093676
4          sacks    0.081707
1       attempts    0.081502
0    completions    0.023600
```

[8]:
```
# Run Model w/ Different Predicted Values

conn = sqlite3.connect('nfl.db')
query = "SELECT * FROM PlayerStats"
player_stats_df = pd.read_sql_query(query, conn)
conn.close()

qb_data = player_stats_df[player_stats_df['position'] == 'QB']

# 1: Average Stats from All Games
```

```
mahomes_avg_stats = qb_data[qb_data['player_display_name'] == 'Patrick␣
 ↪Mahomes'][features].mean()
mahomes_avg_df = pd.DataFrame([mahomes_avg_stats])
predicted_tds_avg = model.predict(mahomes_avg_df)
print(f"Predicted number of passing touchdowns (Average of all games):␣
 ↪{predicted_tds_avg[0]}")

# 2: Stats from Previous Games Against the Ravens
qb_data_with_opponents = qb_data.copy()
qb_data_with_opponents['opponent'] = qb_data_with_opponents.apply(
    lambda row: row['away_team'] if row['recent_team'] == row['home_team'] else␣
 ↪row['home_team'],
    axis=1
)
mahomes_vs_ravens =␣
 ↪qb_data_with_opponents[(qb_data_with_opponents['player_display_name'] ==␣
 ↪'Patrick Mahomes') & (qb_data_with_opponents['opponent'] == 'BAL')]
mahomes_avg_vs_ravens = mahomes_vs_ravens[features].mean()
mahomes_vs_ravens_df = pd.DataFrame([mahomes_avg_vs_ravens])
predicted_tds_vs_ravens = model.predict(mahomes_vs_ravens_df)
print(f"Predicted number of passing touchdowns (Against Ravens):␣
 ↪{predicted_tds_vs_ravens[0]}")

# 3: Recent Game Stats
recent_games = qb_data[qb_data['player_display_name'] == 'Patrick Mahomes'].
 ↪sort_values(by='week', ascending=False).head(5)
mahomes_recent_stats = recent_games[features].mean()
mahomes_recent_df = pd.DataFrame([mahomes_recent_stats])
predicted_tds_recent = model.predict(mahomes_recent_df)
print(f"Predicted number of passing touchdowns (Recent games):␣
 ↪{predicted_tds_recent[0]}")

# 4: Weighted Average of All Methods
combined_stats = 0.5 * mahomes_recent_stats + 0.3 * mahomes_avg_stats + 0.2 *␣
 ↪mahomes_avg_vs_ravens
combined_stats_df = pd.DataFrame([combined_stats])
predicted_tds_combined = model.predict(combined_stats_df)
print(f"Predicted number of passing touchdowns (Weighted average):␣
 ↪{predicted_tds_combined[0]}")
```

```
Predicted number of passing touchdowns (Average of all games):
1.9296677112579346
Predicted number of passing touchdowns (Against Ravens): 2.5652287006378174
Predicted number of passing touchdowns (Recent games): 1.558880090713501
Predicted number of passing touchdowns (Weighted average): 1.990248203277588
```

[22]:
```python
# XGBoost OG 2 - Predicting QB Performance (yards)
# For backtesting features

conn = sqlite3.connect('nfl.db')
query = "SELECT * FROM PlayerStats"
player_stats_df = pd.read_sql_query(query, conn)
conn.close()

qb_data = player_stats_df[player_stats_df['position'] == 'QB']

# Select relevant features for predicting passing yards
features = ['completions', 'attempts', 'interceptions', 'sacks']
target = 'passing_yards'

# Define X and y
X = qb_data[features]
y = qb_data[target]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

# Create and train the XGBoost regressor model
model = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=100,␣
↪learning_rate=0.1, max_depth=3)
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model's performance
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')

# Feature importance
importance = model.feature_importances_
features_importance_df = pd.DataFrame({'Feature': features, 'Importance':␣
↪importance})
print(features_importance_df.sort_values(by='Importance', ascending=False))
```

```
Mean Squared Error: 1962.9090294206608
         Feature  Importance
0    completions    0.976251
1       attempts    0.014337
3          sacks    0.005763
2  interceptions    0.003649
```

```
[10]: # Run Model w/ Different Predicted Values

      conn = sqlite3.connect('nfl.db')
      query = "SELECT * FROM PlayerStats"
      player_stats_df = pd.read_sql_query(query, conn)
      conn.close()

      qb_data = player_stats_df[player_stats_df['position'] == 'QB']

      # 1: Average Stats from All Games
      mahomes_avg_stats = qb_data[qb_data['player_display_name'] == 'Patrick␣
       ↪Mahomes'][features].mean()
      mahomes_avg_df = pd.DataFrame([mahomes_avg_stats])
      predicted_yards_avg = model.predict(mahomes_avg_df)
      print(f"Predicted number of passing yards (Average of all games):␣
       ↪{predicted_yards_avg[0]}")

      # 2: Stats from Previous Games Against the Ravens
      qb_data_with_opponents = qb_data.copy()
      qb_data_with_opponents['opponent'] = qb_data_with_opponents.apply(
          lambda row: row['away_team'] if row['recent_team'] == row['home_team'] else␣
       ↪row['home_team'],
          axis=1
      )
      mahomes_vs_ravens =␣
       ↪qb_data_with_opponents[(qb_data_with_opponents['player_display_name'] ==␣
       ↪'Patrick Mahomes') & (qb_data_with_opponents['opponent'] == 'BAL')]
      mahomes_avg_vs_ravens = mahomes_vs_ravens[features].mean()
      mahomes_vs_ravens_df = pd.DataFrame([mahomes_avg_vs_ravens])
      predicted_yards_vs_ravens = model.predict(mahomes_vs_ravens_df)
      print(f"Predicted number of passing yards (Against Ravens):␣
       ↪{predicted_yards_vs_ravens[0]}")

      # 3: Recent Game Stats
      recent_games = qb_data[qb_data['player_display_name'] == 'Patrick Mahomes'].
       ↪sort_values(by='week', ascending=False).head(5)
      mahomes_recent_stats = recent_games[features].mean()
      mahomes_recent_df = pd.DataFrame([mahomes_recent_stats])
      predicted_yards_recent = model.predict(mahomes_recent_df)
      print(f"Predicted number of passing yards (Recent games):␣
       ↪{predicted_yards_recent[0]}")

      # 4: Weighted Average of All Methods
      combined_stats = 0.5 * mahomes_recent_stats + 0.3 * mahomes_avg_stats + 0.2 *␣
       ↪mahomes_avg_vs_ravens
      combined_stats_df = pd.DataFrame([combined_stats])
```

```
predicted_yards_combined = model.predict(combined_stats_df)
print(f"Predicted number of passing yards (Weighted average):␣
  ↪{predicted_yards_combined[0]}")
```

```
Predicted number of passing yards (Average of all games): 275.55792236328125
Predicted number of passing yards (Against Ravens): 314.8551025390625
Predicted number of passing yards (Recent games): 297.7343444824219
Predicted number of passing yards (Weighted average): 298.9148254394531
```

Model Results:

- **Mean Squared Error (MSE)**: The MSE for the model on the test set is approximately **0.73**. This indicates the average squared difference between the actual and predicted passing touchdowns.
- **Feature Importance**: The importance of each feature used in the model is as follows:
    1. **passing_yards**: 0.710 (most influential)
    2. **interceptions**: 0.090
    3. **sacks**: 0.088
    4. **attempts**: 0.086
    5. **completions**: 0.026 (least influential)

Interpretation:

- **Passing Yards** is the most important feature in predicting the number of passing touchdowns, which aligns with expectations.
- **Interceptions** and **Sacks** also play significant roles, likely because avoiding turnovers and negative plays can directly influence touchdown potential.
- **Attempts** and **Completions** are less influential compared to passing yards but still contribute to the model.

[ ]:

```

```

[ ]:

```

```

[ ]:

```

```

[ ]:

```

```

[ ]:

```

```

Train Model for Predicting Passing Touchdowns

[18]:
```
# Train Model for Predicting Passing Touchdowns

conn = sqlite3.connect('nfl.db')
player_stats_query = "SELECT * FROM PlayerStats"
player_stats_df = pd.read_sql_query(player_stats_query, conn)
conn.close()

qb_data = player_stats_df[player_stats_df['position'] == 'QB']

# Select relevant features for predicting passing touchdowns
features = ['completions', 'attempts', 'passing_yards', 'interceptions',␣
 ↪'sacks']
target_tds = 'passing_tds'

# Define X and y for training the model
X_tds = qb_data[features]
y_tds = qb_data[target_tds]

# Step 3: Train the model for predicting passing touchdowns
X_train_tds, X_test_tds, y_train_tds, y_test_tds = train_test_split(X_tds,␣
 ↪y_tds, test_size=0.2, random_state=42)
model_tds = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=100,␣
 ↪learning_rate=0.1, max_depth=3)
model_tds.fit(X_train_tds, y_train_tds)
```

[18]:
```
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=3, 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, …)
```

Train Model for Predicting Passing Yards

[19]:
```
conn = sqlite3.connect('nfl.db')
player_stats_query = "SELECT * FROM PlayerStats"
player_stats_df = pd.read_sql_query(player_stats_query, conn)
conn.close()

qb_data = player_stats_df[player_stats_df['position'] == 'QB']

# Select relevant features for predicting passing yards
```

```
features = ['completions', 'attempts', 'passing_yards', 'interceptions',␣
 ↪'sacks']
target_yards = 'passing_yards'

# Define X and y for training the model
X_yards = qb_data[features]
y_yards = qb_data[target_yards]

# Step 3: Train the model for predicting passing yards
X_train_yards, X_test_yards, y_train_yards, y_test_yards =␣
 ↪train_test_split(X_yards, y_yards, test_size=0.2, random_state=42)
model_yards = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=100,␣
 ↪learning_rate=0.1, max_depth=3)
model_yards.fit(X_train_yards, y_train_yards)
```

[19]:
```
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=3, 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, …)
```

[20]:
```
# Predict Passing Touchdowns for a Specific QB

# Set the quarterback's name
qb_name = 'Patrick Mahomes'  # Change this to the name of any quarterback

# Filter the data for the specific quarterback
qb_specific_data = qb_data[qb_data['player_display_name'] == qb_name]

# Prepare the features for the upcoming game based on the specific QB's␣
 ↪historical data
X_upcoming_game_tds = qb_specific_data[features]

# Use the trained model to predict passing touchdowns
predicted_tds = model_tds.predict(X_upcoming_game_tds)
predicted_tds_avg = predicted_tds.mean()

# Print the predictions
print(f"Predicted Passing Touchdowns for {qb_name}: {predicted_tds_avg}")
```

```
Predicted Passing Touchdowns for Patrick Mahomes: 1.94794499874115
```

[21]:
```
# Predict Passing Touchdowns for a Specific QB

# Set the quarterback's name
qb_name = 'Patrick Mahomes'  # Change this to the name of any quarterback

# Filter the data for the specific quarterback
qb_specific_data = qb_data[qb_data['player_display_name'] == qb_name]

# Prepare the features for the upcoming game based on the specific QB's␣
 ↪historical data
X_upcoming_game_tds = qb_specific_data[features]

# Use the trained model to predict passing touchdowns
predicted_tds = model_yards.predict(X_upcoming_game_tds)
predicted_tds_avg = predicted_tds.mean()

# Print the predictions
print(f"Predicted Passing Touchdowns for {qb_name}: {predicted_tds_avg}")
```

```
Predicted Passing Touchdowns for Patrick Mahomes: 294.4209289550781
```

[ ]:
```

```