Joe Stella

Progress Report 2

import xgboost as xgb

import pandas as pd

from sklearn.preprocessing import LabelEncoder

def load\_model\_and\_predict(batter\_id, pitcher\_id, batter\_stats, pitcher\_stats):

    # Load the saved XGBoost model

    batter\_performance\_model = xgb.XGBClassifier()

    batter\_performance\_model.load\_model("batter\_performance\_xgb\_model.xgb")

    # Get the batter and pitcher statistics

    batter\_data = batter\_stats[batter\_stats['last\_name, first\_name'] == batter\_id]

    pitcher\_data = pitcher\_stats[pitcher\_stats['last\_name, first\_name'] == pitcher\_id]

    if batter\_data.empty or pitcher\_data.empty:

        print("Invalid batter or pitcher ID provided. Please try again.")

        return

    # Prepare the feature data

    X\_test = pd.DataFrame({

        'woba\_x': batter\_data['woba'],

        'woba\_y': pitcher\_data['woba'],

        'k\_percent\_x': batter\_data['k\_percent'],

        'k\_percent\_y': pitcher\_data['k\_percent'],

        'whiff\_percent\_x': batter\_data['whiff\_percent'],

        'whiff\_percent\_y': pitcher\_data['whiff\_percent'],

        'swing\_percent\_x': batter\_data['swing\_percent'],

        'swing\_percent\_y': pitcher\_data['swing\_percent'],

        'sweet\_spot\_percent\_x': batter\_data['sweet\_spot\_percent'],

        'sweet\_spot\_percent\_y': pitcher\_data['sweet\_spot\_percent'],

        'hard\_hit\_percent\_x': batter\_data['hard\_hit\_percent'],

        'hard\_hit\_percent\_y': pitcher\_data['hard\_hit\_percent'],

        'avg\_best\_speed\_x': batter\_data['avg\_best\_speed'],

        'avg\_best\_speed\_y': pitcher\_data['avg\_best\_speed']

    }, index=[0])

    # Make predictions

    y\_pred = batter\_performance\_model.predict(X\_test)

    # Decode the predicted labels back to their original string values

    label\_encoder = LabelEncoder()

    label\_encoder.classes\_ =['strikeout', 'single', 'field\_out', 'walk', 'double',

    'grounded\_into\_double\_play', 'sac\_fly', 'caught\_stealing\_2b',

    'fielders\_choice', 'triple', 'force\_out', 'home\_run', 'double\_play',

    'field\_error', 'hit\_by\_pitch', 'fielders\_choice\_out',

    'strikeout\_double\_play', 'sac\_bunt', 'catcher\_interf', 'sac\_fly\_double\_play',

    'pickoff\_1b', 'triple\_play', 'other\_out', 'caught\_stealing\_3b',

    'pickoff\_caught\_stealing\_2b', 'caught\_stealing\_home']  # Provide the list of label names

    y\_pred\_decoded = label\_encoder.inverse\_transform(y\_pred)

    return y\_pred\_decoded[0]

if \_\_name\_\_ == "\_\_main\_\_":

    # Load the batter and pitcher statistics

    batter\_stats = pd.read\_csv("batterstats.csv")

    pitcher\_stats = pd.read\_csv("pitcherstats.csv")

    # Prompt the user for batter and pitcher IDs

    batter\_id = input("Enter the batter ID: ")

    pitcher\_id = input("Enter the pitcher ID: ")

    print("Batter player\_id values:")

    print(batter\_stats['player\_id'].unique())

    print("\nPitcher player\_id values:")

    print(pitcher\_stats['player\_id'].unique())

    print(pitcher\_stats['player\_id'] == pitcher\_id)

    # Load the model and make predictions

    prediction = load\_model\_and\_predict(batter\_id, pitcher\_id, batter\_stats, pitcher\_stats)

    if prediction:

        print(f"Predicted batter performance: {prediction}")

This week, I made significant progress on the prediction method for our project. I worked hard to implement a robust mechanism that uses the trained model to predict outcomes based on the provided batter and pitcher data. Alongside refining the prediction algorithm, I spent time enhancing the model's accuracy by adding more statistical features to the dataset. This comprehensive approach aims to ensure that my predictions are precise and reflective of the dynamic nuances of the game.

In a strategic move to further optimize our predictive capabilities, I decided to transition from using XGBRegressor to XGBClassifier. This shift was motivated by the classifier's suitability for predicting discrete classes, aligning better with our project's requirements. However, this transition came with its challenges. During the train-test split process, I encountered a discrepancy between the expected and inferred classes, resulting in a significant obstacle to successful model training.

The ValueError, indicating inconsistencies in the inferred classes, underscores the complexity of adapting our model to the new classifier. Resolving this issue is crucial to restoring the integrity of our training process and advancing towards our predictive goals. In the upcoming week, my immediate focus will be on diagnosing and rectifying this error to ensure uninterrupted model training.

Looking ahead, my roadmap includes resolving the current obstacle and embarking on crucial validation steps. Once the model training is back on track, my next priority is to rigorously test its predictive accuracy through real-life observations. This empirical validation will provide invaluable insights into the model's efficacy and guide any necessary adjustments to the underlying statistical features.

Furthermore, I plan to enrich our predictive framework by introducing the capability to rank outcomes based on their likelihood. This enhancement will offer users a nuanced understanding of the relative probabilities associated with different game scenarios, empowering them to make informed decisions based on the predictive insights provided.

In summary, while I encountered a significant challenge this week, I remain committed to overcoming obstacles and advancing our project towards its ultimate objectives. With a clear roadmap and a resilient spirit, I am poised to navigate the complexities of model optimization and validation, ensuring that our predictive tool emerges as a reliable asset in the realm of sports analytics.