Football Match Probability Prediction

Ian Burns, Gabriel Carson, Ethan Head

Dataset/Data Collection

- The data set contains more than 150000 historical world football matches from 2019 to 2021, with more than 860 leagues and 9500 teams.
- You are given train.csv and test.csv. Train.csv has a target column that specifies the winner [home, away, draw], while test.csv doesn’t have a target column.
- Your goal is to predict the probabilities for each match outcome in the target columns: the probabilities of home to win, draw, and away to win.
- For a submission, you build your model using data from train.csv, then make predictions for test.csv and submit that file.

302.14 MB
189 columns
150,000 samples

Feature Extraction

Features we wanted: Historical columns

Historical home team features:

home_team_history_match_date_{i} - The date of the last i-th match played by Home team.
home_team_history_is_play_home_{i} - If 1, the Home team played home.
home_team_history_is_cup_{i} - If 1, the match was a cup competition.
home_team_history_goal_{i} - The number of goals scored by the Home team on its last i-th match.
home_team_history_opponent_goal_{i} - The number of goals conceded by the Home team on its last i-th match.
home_team_history_rating_{i} - The rating of the Home team on its last i-th match (pre match rating).
home_team_history_opponent_rating_{i} - The rating of the opponent team on Home team last i-th match (pre match rating).
home_team_history_coach_{i} - The coach id of the Home team on its last i-th match.
home_team_history_league_id_{i} - The league name id by the Home team on its last i-th match.

Historical away team features match the home team features, but have ‘away’ as the leading term

Feature Extraction

Features we dropped: Descriptive columns

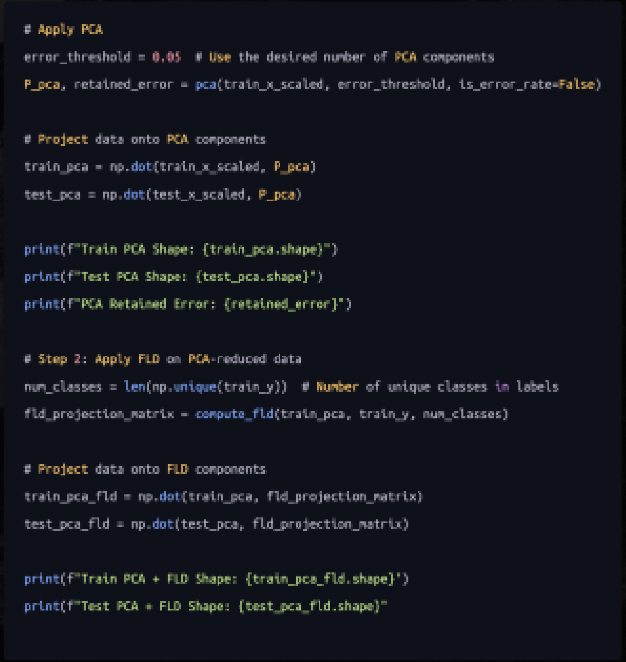
target - The variable you have to predict the probabilities only available in the train set.
home_team_name - The name of the Home the team. Hidden in test set, see this discussion
away_team_name - The name of the Away the team. Hidden in test set, see this discussion
match_date - The match date (UTC).
league_name - The league name.
league_id - The league id. Note that league names can be identical for two differents id.
is_cup - If the value is 1 the match is played for a cup competition.
home_team_coach_id - The id of the Home team coach.
away_team_coach_id - The id of the Away team coach.

Some of these columns are strings that aren't worth encoding, some have hashed ids, and one has a low-correlation boolean value.

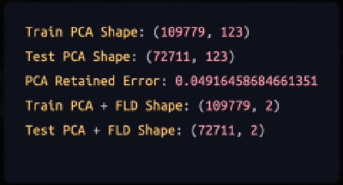
Feature Reduction

- PCA - linear dimensionality reduction technique
    - Reduces the high-dimensional Kaggle football match dataset to essential features.
    - Retains key patterns in match statistics by selecting principal components with minimal error.
    - Prepares the reduced data for processing by eliminating redundant features.
- FLD -
    - Projects the dataset onto a lower-dimensional space optimized for class discrimination.
    - Maximizes separability between match outcomes (e.g., home win, away win, draw) by focusing on inter-class variance.
    - Adjusts projection dimensions based on the number of match outcome classes. (Resulting dimension is [num classes - 1])
- PCA + FLD
    - Perform PCA first, to maintain as much variance as possible, then perform FLD on the result to bring data down to 2D (this dataset has 3 classes)

Feature Reduction

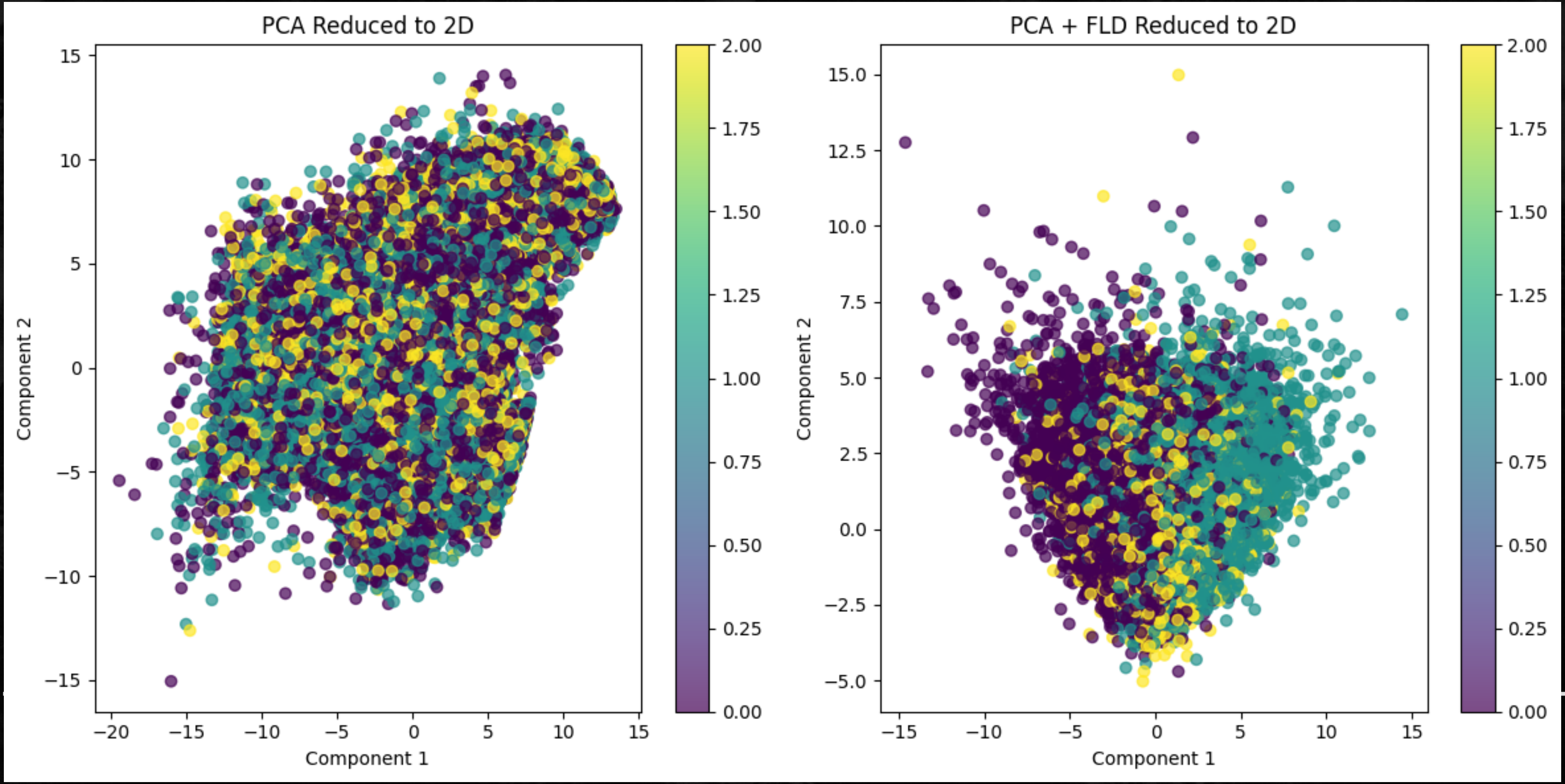


```python
# Apply PCA
error_threshold = 0.05  # Use the desired number of PCA components
P_pca, retained_error = pca(train_x_scaled, error_threshold, is_error_rate=False)

# Project data onto PCA components
train_pca = np.dot(train_x_scaled, P_pca)
test_pca = np.dot(test_x_scaled, P_pca)

print(f"Train PCA Shape: {train_pca.shape}")
print(f"Test PCA Shape: {test_pca.shape}")
print(f"PCA Retained Error: {retained_error}")

# Step 2: Apply FLD on PCA-reduced data
num_classes = len(np.unique(train_y))  # Number of unique classes in labels
fld_projection_matrix = compute_fld(train_pca, train_y, num_classes)

# Project data onto FLD components
train_pca_fld = np.dot(train_pca, fld_projection_matrix)
test_pca_fld = np.dot(test_pca, fld_projection_matrix)

print(f"Train PCA + FLD Shape: {train_pca_fld.shape}")
print(f"Test PCA + FLD Shape: {test_pca_fld.shape}"
```

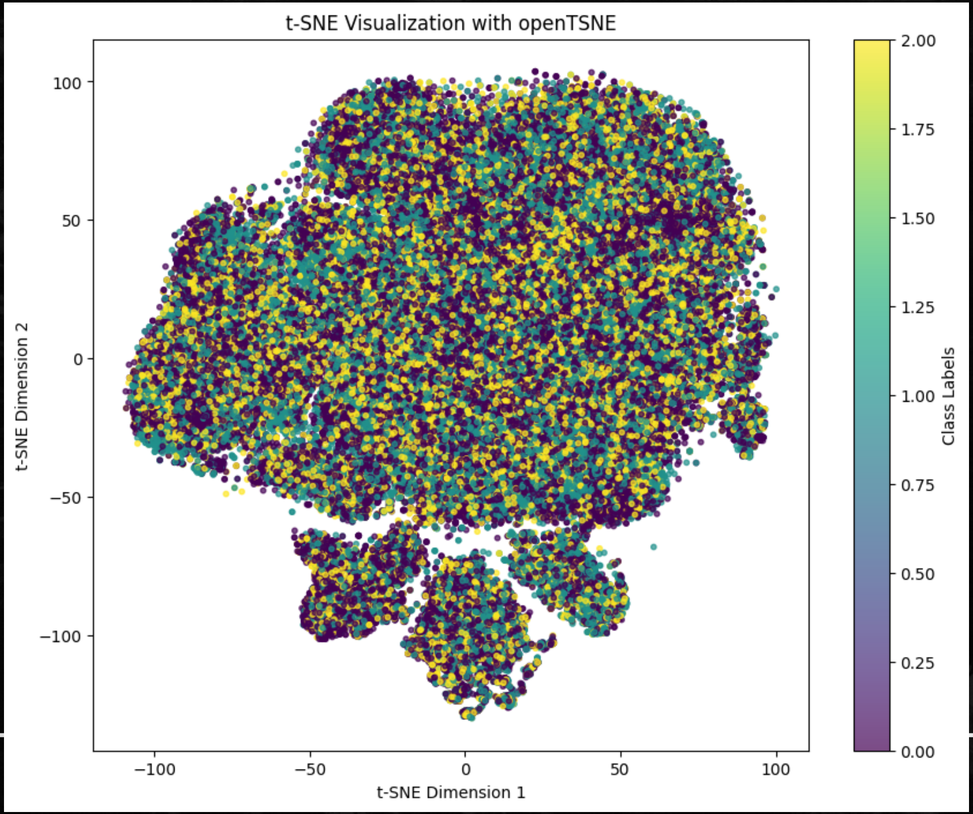


```
Train PCA Shape: (109779, 123)
Test PCA Shape: (72711, 123)
PCA Retained Error: 0.04916458684661351
Train PCA + FLD Shape: (109779, 2)
Test PCA + FLD Shape: (72711, 2)
```

PCA + FLD Results



t-SNE


t-SNE Visualization with openTSNE
t-SNE Dimension 1
t-SNE Dimension 2
Class Labels
100
50
0
−50
−100
−100
−50
0
50
100
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00

Classification/Regression

- MPP
    - Implements MPP classification to compute class probabilities for test samples.
    - Supports Euclidean and Mahalanobis metrics by calculating distances to class means.
    - Uses class-specific means and covariances (for Mahalanobis) for precise distance measurement.
    - Converts distances into normalized probabilities for each class.

- HAC (Hierarchical Agglomerative Clustering)
    - Applies Hierarchical Agglomerative Clustering (HAC) with Ward linkage to a fraction of training data.
    - Uses sampled data to compute cluster-to-class probabilities based on training labels.
    - Processes test data in batches, assigning probabilities by finding nearest training clusters.
    - Supports scalability with adjustable batch size and data fraction

Classification/Regression

- Neural Network
  - Initializes a multi-layer neural network with specified layer sizes, random weights, and biases.
  - Trains the network using mini-batch Stochastic Gradient Descent (SGD) with backpropagation to update weights and biases.
  - Predicts outputs by passing input data through the network using forward propagation.
  - Assesses performance on test data by comparing predicted and true labels.

- Random Forest
  - Trains a Random Forest model using k-fold cross-validation to compute the mean log loss.
  - Evaluates the trained model on a separate validation set, calculating log loss for predictions.
  - Utilizes parallel processing (n_jobs=-1) for faster training.
  - Uses fit_predict_random_forest to train and predict probabilities during cross-validation.

Fusion

BKS

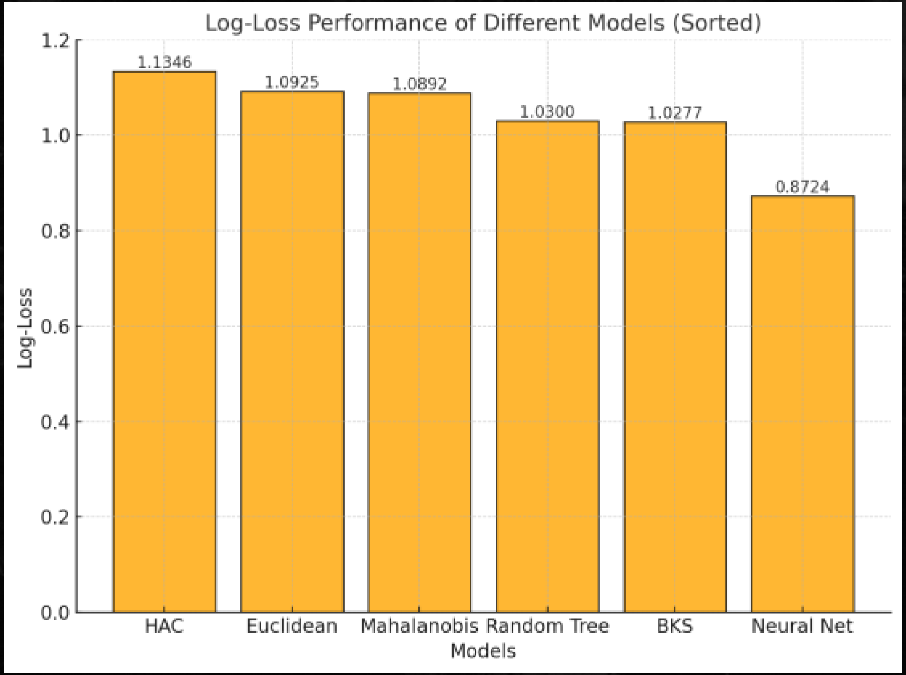
- Combines predictions from the MPP Minimum distance models to improve match outcome accuracy.
- Creates a lookup table by pairing predictions from two classifiers with the true match outcome during training.
- Uses the BKS table during testing to predict outcomes based on model agreement or the most likely true label.
- Leverages complementary strengths of different models to handle complex match data patterns more effectively.

Performance Evaluation

- Log-Loss
  - Evaluates the accuracy of predicted match outcome probabilities on the dataset.
  - Penalizes confident but incorrect predictions more heavily, encouraging probabilistic accuracy.
- K-fold
  - Ensures robust evaluation of clustering performance on the Kaggle football dataset by splitting data into training and test folds.
  - Trains hierarchical agglomerative clustering on different data subsets in each fold to prevent overfitting.
  - Averages performance metrics across all folds to provide a reliable model evaluation.

Performance Evaluation Comparison

- 1st - Neural Net
- 2nd - BKS (MPP)
- 3rd - Random Tree



Q&A