Player Position Classifier based on Stats

Vinh Tran

Spring ‘24

Purpose and Metadata

- To predict the primary position of soccer players based on their attributes
- EA Sports FC 24 complete datasets
  - Specifically, ‘female_players.csv’ and ‘male_players.csv’
- Both datasets had 108 columns
  - Female players dataset had 5,035 rows
  - Male players dataset had 180,021 rows

Data Cleaning

- **Data Summary:**
  - **Rows: 185,056 (after cleaning)**
  - **Columns: 7 (after cleaning)**
- **Data Cleaning:**
  - **Dropped unnecessary columns for the purpose from both datasets**
- **Concatenate**

```
players.head()
```

	player_positions	pace	shooting	passing	dribbling	defending	physic
0	ST, LW	97.0	90.0	80.0	92.0	36.0	78.0
1	ST	89.0	93.0	66.0	80.0	45.0	88.0
2	CM, CAM	72.0	88.0	94.0	87.0	65.0	78.0
3	CF, CAM	80.0	87.0	90.0	94.0	33.0	64.0
4	CF, ST	79.0	88.0	83.0	87.0	39.0	78.0

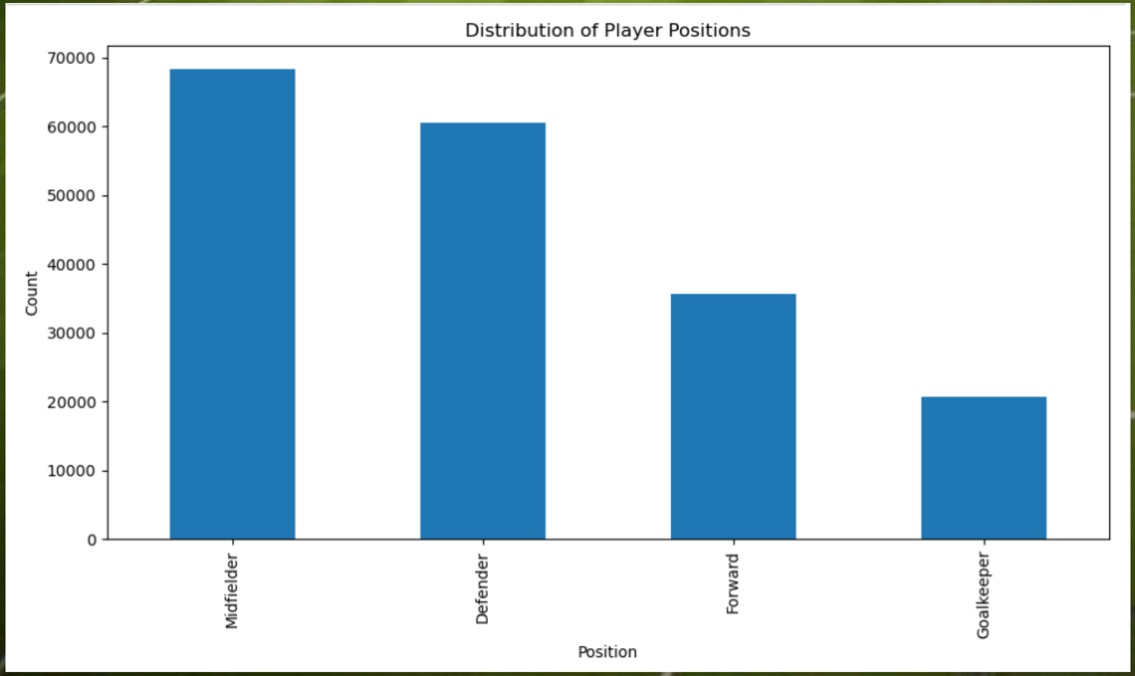
Data Preparation

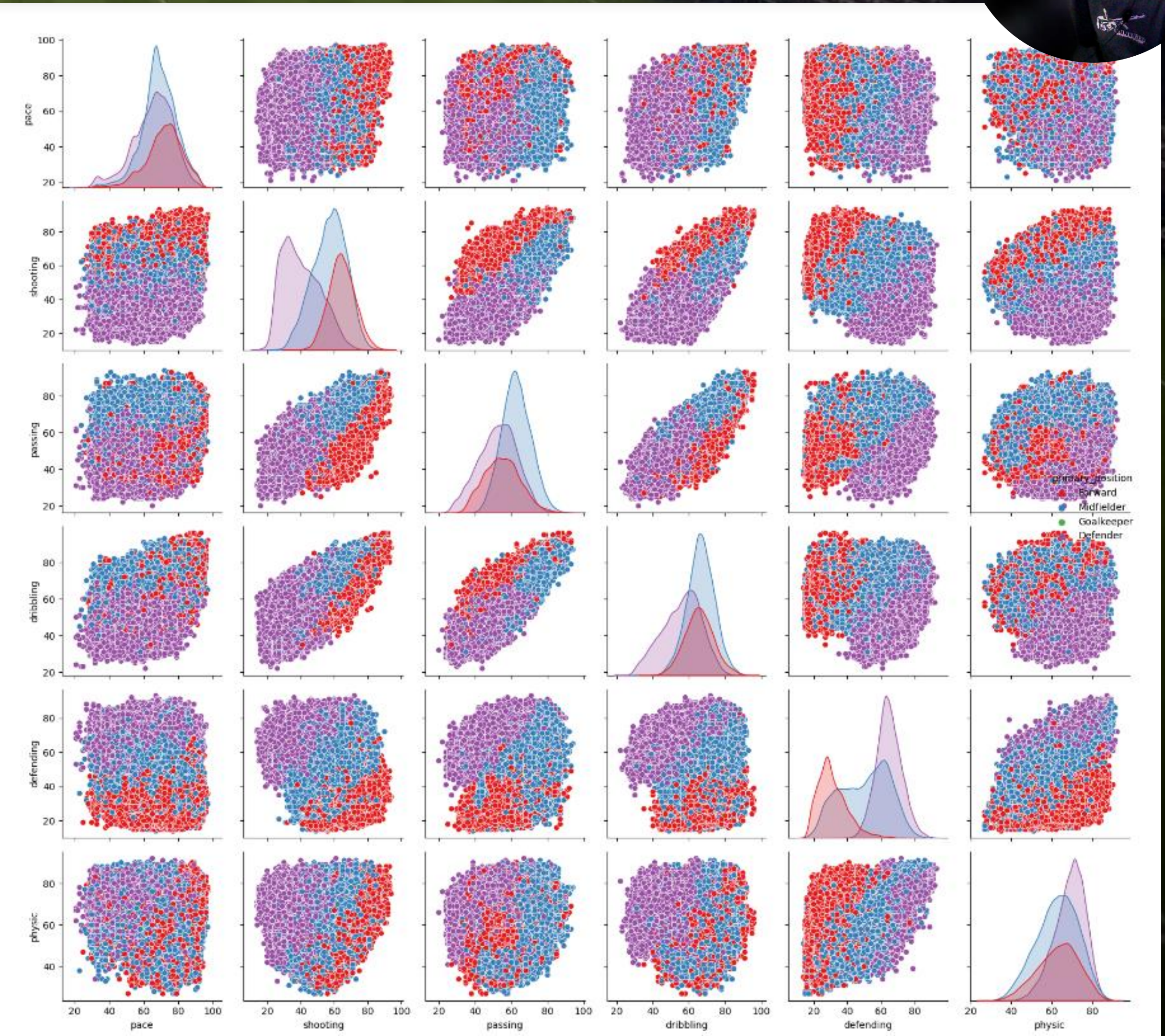
- Too many unique values in ‘player_positions’ column
- Simplified into primary categories
  - Forward – ST, CF, LW, and RW
  - Midfielder – CM, CAM,CDM, LM, and RM
  - Defender – CB, LB, RB, LWB, and RWB
  - Goalkeeper
- ‘player_positions’ also listed secondary position
  - Only used their primary position

```python
# Simplify positions into primary categories
position_mapping = {
    'ST': 'Forward', 'CF': 'Forward', 'LW': 'Forward', 'RW': 'Forward',
    'CM': 'Midfielder', 'CAM': 'Midfielder', 'CDM': 'Midfielder', 'LM': 'Midfielder', 'RM': 'Midfielder',
    'CB': 'Defender', 'LB': 'Defender', 'RB': 'Defender', 'LWB': 'Defender', 'RWB': 'Defender',
    'GK': 'Goalkeeper'
}
```

```python
# Apply mapping
players['primary_position'] = players['player_positions'].apply(lambda x: position_mapping.get(x.split(',')[0], 'Other'))

# Filter out 'Other' positions for simplicity
players = players[players['primary_position'] != 'Other']
```

Data Visualizations





Pair Plot of Features Colored by Position

Feature Engineering

- **Selected Features:**
  - **Pace**
  - **Shooting**
  - **Passing**
  - **Dribbling**
  - **Defending**
  - **Physical**

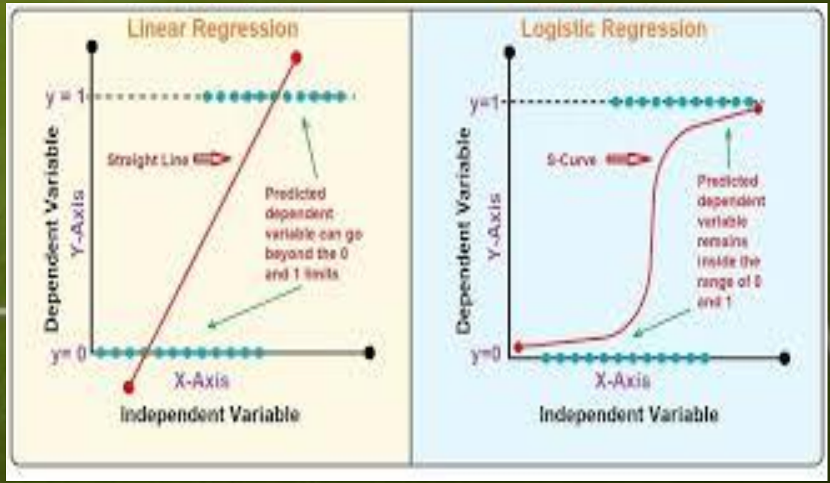
- **Target Variable – Primary position of players**
- **Data Preprocessing:**
  - **Filled missing values with the mean**
  - **Split data (80% training, 20% testing)**
  - **Normalized features using StandardScaler**

```python
# Step 3: Feature Selection/Engineering
# Feature Selection/Engineering
features = ['pace', 'shooting', 'passing', 'dribbling', 'defending', 'physic']
X = players[features]
y = players['primary_position']
X.fillna(X.mean(), inplace=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Normalize/Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

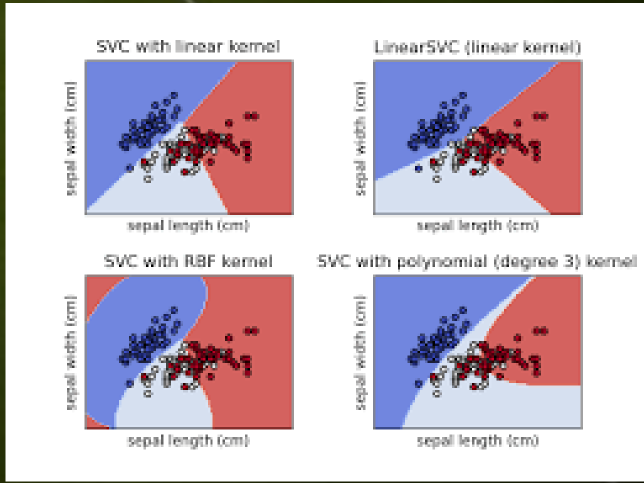
Model Selection & Justificatio

- **Models Tried:**
  - **K-Nearest Neighbors – Simple, non-parametric method**
  - **Logistic Regression – Good baseline for classification**
  - **SVM (linear kernel) – Effective in high-dimensional spaces**

K-Nearest Neighbor



Logistic Regression



Support Vector Machine

Parameter Tuning & Resul

- **KNN: Selected number of neighbors (k=5) through experimentation.**
- **Logistic Regression: Maximum iterations set to 1000.**
- **SVM: Used linear kernel for interpretability.**

```
# K-Nearest Neighbors
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
knn_pred = knn.predict(X_test)
print("KNN Accuracy:", accuracy_score(y_test, knn_pred))
print(classification_report(y_test, knn_pred))
```

```
KNN Accuracy: 0.858208148708527
              precision    recall  f1-score   support

    Defender       0.89      0.89      0.89     12163
     Forward       0.85      0.79      0.82      7195
  Goalkeeper       1.00      1.00      1.00      4182
  Midfielder       0.80      0.82      0.81     13472

    accuracy                           0.86     37012
   macro avg       0.88      0.88      0.88     37012
weighted avg       0.86      0.86      0.86     37012
```

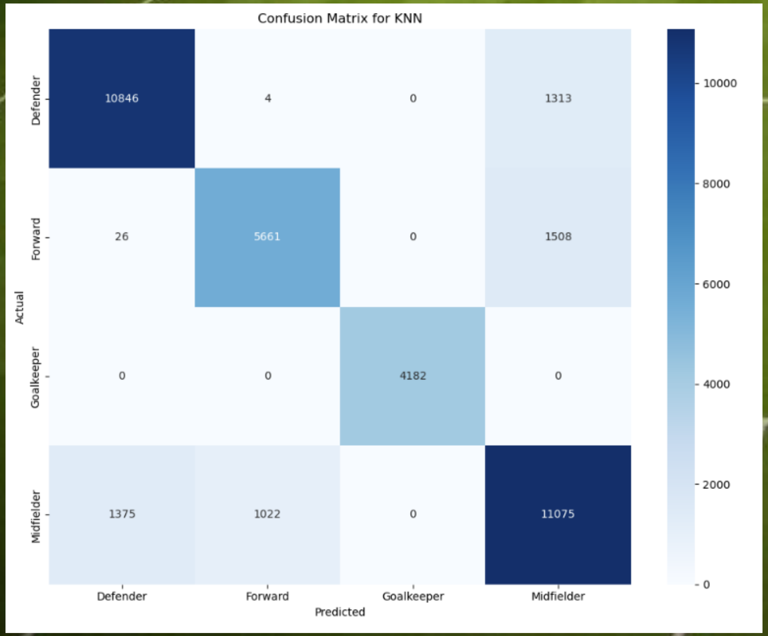
```
# Logistic Regression
log_reg = LogisticRegression(max_iter=1000)
log_reg.fit(X_train, y_train)
log_reg_pred = log_reg.predict(X_test)
print("Logistic Regression Accuracy:", accuracy_score(y_test, log_reg_pred))
print(classification_report(y_test, log_reg_pred))
```

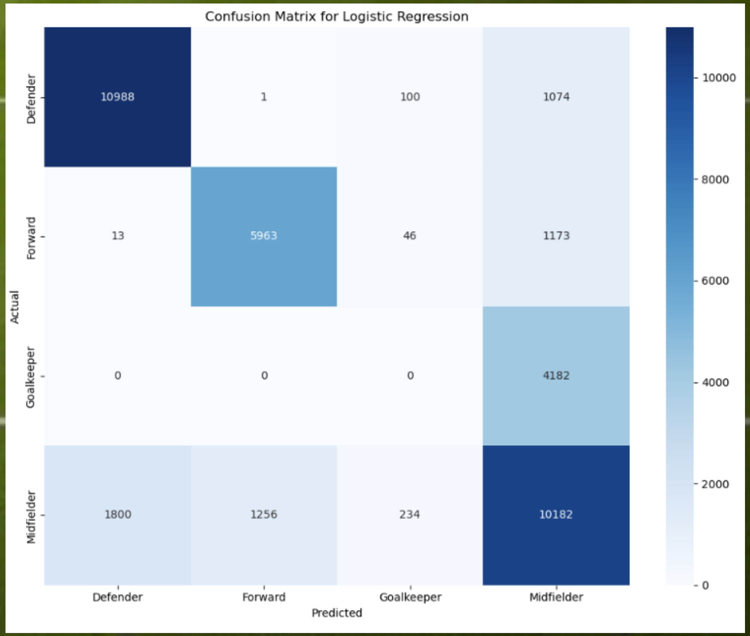
```
Logistic Regression Accuracy: 0.7330865665189669
              precision    recall  f1-score   support

    Defender       0.86      0.90      0.88     12163
     Forward       0.83      0.83      0.83      7195
  Goalkeeper       0.00      0.00      0.00      4182
  Midfielder       0.61      0.76      0.68     13472

    accuracy                           0.73     37012
   macro avg       0.57      0.62      0.60     37012
weighted avg       0.67      0.73      0.70     37012
```

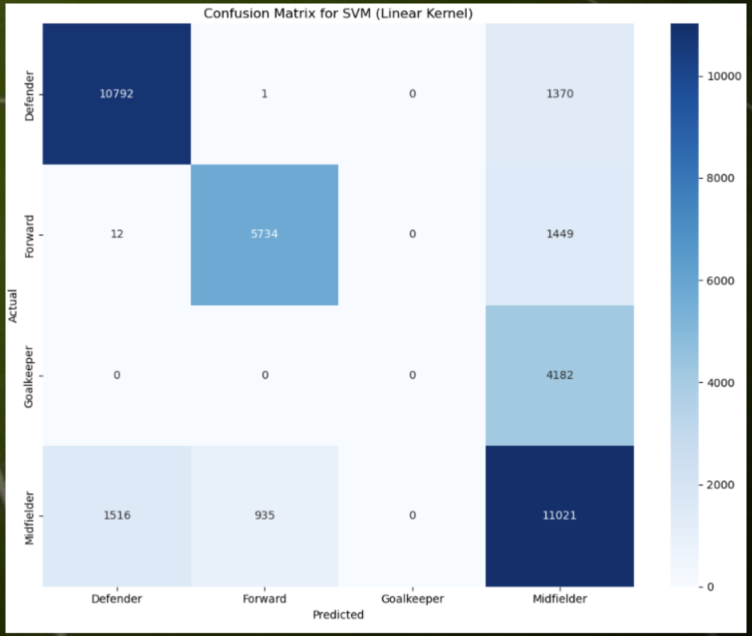
```
# Support Vector Machine
svm = SVC(kernel='linear',probability=True)
svm.fit(X_train, y_train)
svm_pred = svm.predict(X_test)
print("SVM Accuracy:", accuracy_score(y_test, svm_pred))
print(classification_report(y_test, svm_pred))
```

```
SVM Accuracy: 0.7442721279585

C:\Users\vinht\anaconda3\Lib\site-packages\sklearn\metrics\_
re are ill-defined and being set to 0.0 in labels with no pr
vior.
  _warn_prf(average, modifier, msg_start, len(result))

              precision    recall  f1-score   support

    Defender       0.88      0.89      0.88     12163
     Forward       0.86      0.80      0.83      7195
  Goalkeeper       0.00      0.00      0.00      4182
  Midfielder       0.61      0.82      0.70     13472

    accuracy                           0.74     37012
   macro avg       0.59      0.63      0.60     37012
weighted avg       0.68      0.74      0.71     37012
```

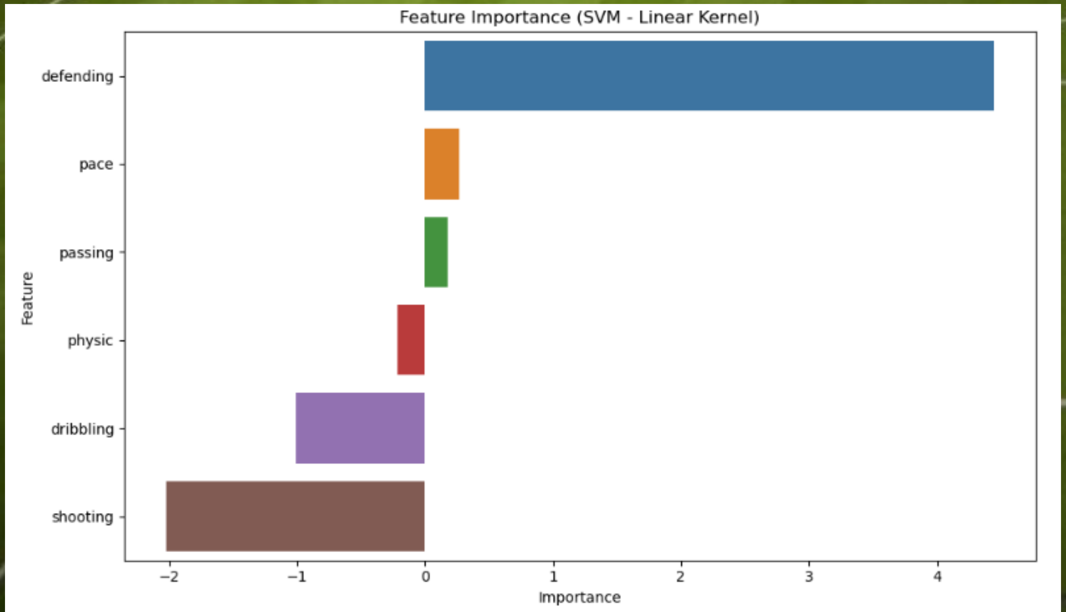
Predictive Errors

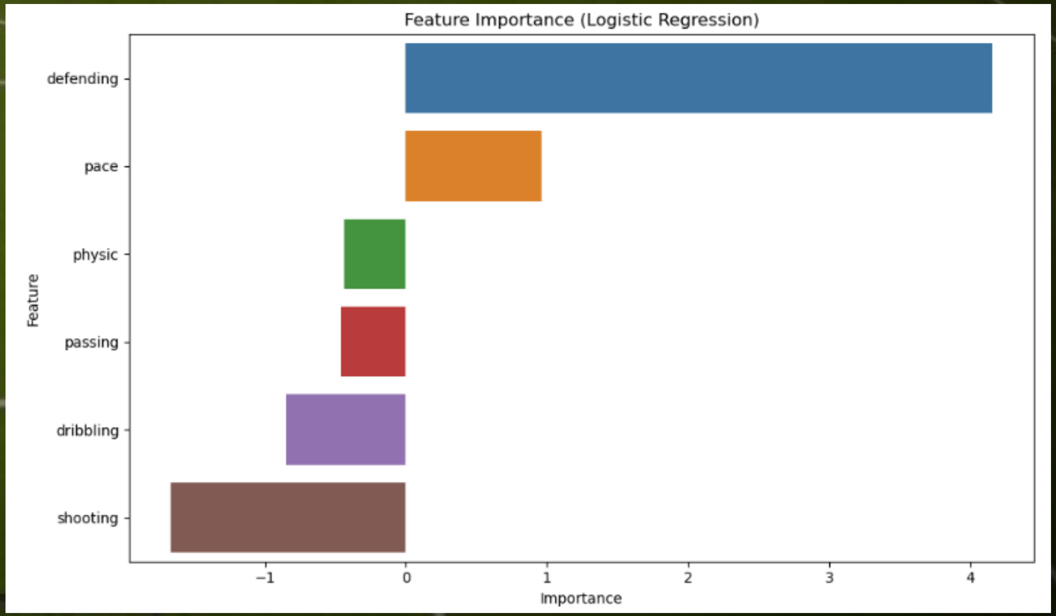






Feature Importance





Conclusion

- **KNN model had the highest percentage of accuracy (85.6%) and was able to correctly classify Goalkeepers**
- **The feature ‘defending’ is highly influential for certain positions according to the Logistic Regression and SVM models**

Experience