technical_assessment_blue_jays

September 7, 2023

1 Toronto Blue Jays Assessment

1.0.1 Import Packages

[1]:
```
# Import Packages
import pandas as pd
import numpy as np
```

1.0.2 Import Datasets

[2]:
```
# Import Data Sets
df_training = pd.read_csv('training.csv')
```

1.0.3 Filter Data Sets

- Drop all rows containing NULL Values.
- Since we are working with fastballs, I have filtered all pitches that are slower than 85 mph

[3]:
```
# Drop all rows that include NULL Values
df_training = df_training.dropna()

# Filter out pitches slower than 85 mph
df_training = df_training[df_training['Velo'] >= 85]
```

1.0.4 Machine Learning Model Creation

- I decided to use a classification model because I was dealing with a binary target, "InPlay".
- Since the task was to predict the probability of a pitch being put into play, I decided to use a logistic regression model.
- This type of model can estimate the probability that pitch will be put into play given the inputted features and will always provide an output between 0 and 1

[4]:
```
# Import necessary libraries
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

# X is Features, y is target
X = df_training[['Velo','SpinRate','HorzBreak','InducedVertBreak']]
```

```
y = df_training.InPlay

# 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=0)

# Initialize and train the Logistic Regression model
logistic_model = LogisticRegression(random_state=0)
logistic_model.fit(X_train, y_train)

# Predict the probability of each pitch being put in play, 1 = Put into play
y_probabilities = logistic_model.predict_proba(X_test)[:, 1]
```

1.0.5 Predicting Balls In Play

- We need to read in 'deploy.csv'

Read in Dataset and apply filtering similar to training dataset

[5]:
```
# Import 'deploy.csv'
df_deploy = pd.read_csv('deploy.csv')
# Drop rows containing NULL Values
df_deploy = df_deploy.dropna()
# Filter pitches thrown slower than 85 mph
df_deploy = df_deploy[df_deploy['Velo'] >= 85]
```

Predict Balls in Play in 'deploy.csv' using its features and the Model created

[6]:
```
# Predict Balls in Play
df_deploy['InPlay'] = logistic_model.predict_proba(df_deploy)[:, 1]
```

Write new csv to directory

[7]:
```
df_deploy.to_csv('deploy_new.csv')
```

Functions and Dictionaries to help with Plotting

[8]:
```
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors

# Function to get colour based on value
def get_color(value, cmap_name='coolwarm', vmin=None, vmax=None):
    # Create a colormap object
    cmap = plt.get_cmap(cmap_name)

    # Normalize the value to the range [0, 1] using vmin and vmax
    norm = mcolors.Normalize(vmin=vmin, vmax=vmax)
```

```
    # Map the normalized value to a color using the colormap
    color = cmap(norm(value))

    # Convert the RGBA color tuple to a hex color code
    hex_color = mcolors.rgb2hex(color)

    return hex_color

# Dictionary to plot labels
columns_dict={
'Velo':'Velocity (mph)',
'SpinRate':'Spin Rate (rpm)',
'HorzBreak':'Horizontal Break (in)',
'InducedVertBreak':'Induced Vertical Break (in)'}

# Dictionary to plot titles
columns_dict_title={
'Velo':'Velocity',
'SpinRate':'Spin Rate',
'HorzBreak':'Horizontal Break',
'InducedVertBreak':'Induced Vertical Break',}
```

Plotting of Balls in Play vs Features

[9]:
```
import seaborn as sns
sns.set_theme(style="whitegrid", palette="pastel")
fig, axs = plt.subplots(4, 4, figsize=(20, 20),dpi=600)
# fig.set_facecolor('white')
for i in range(0,4):
    for j in range(0,4):
        ax = axs[i,j]
        if i == j:
            # Create a histogram for variables along the diagonal
            data = df_deploy[df_deploy.columns[i]]
            cm = plt.cm.get_cmap('coolwarm')
            Y,X = np.histogram(data, 25, normed=1)
            x_span = X.max()-X.min()
            C = [cm(((x-X.min())/x_span)) for x in X]

            N, bins, patches = ax.hist(data, edgecolor='white', linewidth=1)

            cmap_name = 'coolwarm'
            vmin = 0
            vmax = 0.5

            for k in range(len(patches)):
```

```
                value = df_deploy[(df_deploy[df_deploy.columns[i]] >=␣
↪bins[k])&(df_deploy[df_deploy.columns[i]] < bins[k+1])]['InPlay'].mean()
                patches[k].set_facecolor(color = get_color(value, cmap_name,␣
↪vmin, vmax))

            norm = plt.Normalize(0,.50)
            sm = plt.cm.ScalarMappable(cmap='coolwarm', norm=norm)
            cbar  = ax.figure.colorbar(sm,␣
↪ax=ax,orientation='vertical',aspect=15,shrink=0.7,label='Probability of In␣
↪Play')

            ax.set_xlabel(f'{columns_dict[df_deploy.columns[i]]}')

            ax.set_title(f'{columns_dict_title[df_deploy.columns[i]]}')

        elif i < j:
            # Create a 2D hexbin plot for upper triangle
            ax.hexbin(data=df_deploy,x=df_deploy.columns[i],y=df_deploy.
↪columns[j], gridsize=20,␣
↪C='InPlay',cmap='coolwarm',edgecolor='black',linewidth=0.3,vmin=0,vmax=.
↪50,mincnt=5)

            norm = plt.Normalize(0,.50)
            sm = plt.cm.ScalarMappable(cmap='coolwarm', norm=norm)
            cbar  = ax.figure.colorbar(sm,␣
↪ax=ax,orientation='vertical',aspect=15,shrink=0.7,label='Probability of In␣
↪Play')

            ax.set_xlabel(f'{columns_dict[df_deploy.columns[i]]}')
            ax.set_ylabel(f'{columns_dict[df_deploy.columns[j]]}')
            ax.set_title(f'{columns_dict_title[df_deploy.columns[i]]} vs␣
↪{columns_dict_title[df_deploy.columns[j]]}')
        else:
            # No plots in the lower triangle
            ax.axis('off')

fig.suptitle('Probability of Ball In Play - Feature Hex Bins - Min. 5 Pitches␣
↪per Bin',x=0.5,y=1,fontsize=36)
fig.tight_layout()
fig.savefig('output.png')
```

```
<ipython-input-9-030b1d492a18>:12: VisibleDeprecationWarning: Passing
`normed=True` on non-uniform bins has always been broken, and computes neither
the probability density function nor the probability mass function. The result
is only correct if the bins are uniform, when density=True will produce the same
result anyway. The argument will be removed in a future version of numpy.
  Y,X = np.histogram(data, 25, normed=1)
```

```
<ipython-input-9-030b1d492a18>:28: MatplotlibDeprecationWarning: Auto-removal of
grids by pcolor() and pcolormesh() is deprecated since 3.5 and will be removed
two minor releases later; please call grid(False) first.
  cbar  = ax.figure.colorbar(sm,
ax=ax,orientation='vertical',aspect=15,shrink=0.7,label='Probability of In
Play')
<ipython-input-9-030b1d492a18>:40: MatplotlibDeprecationWarning: Auto-removal of
grids by pcolor() and pcolormesh() is deprecated since 3.5 and will be removed
two minor releases later; please call grid(False) first.
  cbar  = ax.figure.colorbar(sm,
ax=ax,orientation='vertical',aspect=15,shrink=0.7,label='Probability of In
Play')
```