TV Show Shark Tank with Machine Learning

March 6, 2024

[1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings(action='ignore', category=FutureWarning)
```

1 Reading in the data

[2]:
```
explore_df = pd.read_csv('Shark Tank US dataset.csv')
explore_df.head()
```

[2]:
```
   Season Number Season Start Season End  Episode Number  Pitch Number  \
0              1      9-Aug-09   5-Feb-10               1             1
1              1      9-Aug-09   5-Feb-10               1             2
2              1      9-Aug-09   5-Feb-10               1             3
3              1      9-Aug-09   5-Feb-10               1             4
4              1      9-Aug-09   5-Feb-10               1             5

  Original Air Date              Startup Name           Industry  \
0          9-Aug-09             AvaTheElephant    Health/Wellness
1          9-Aug-09        Mr.Tod'sPieFactory  Food and Beverage
2          9-Aug-09                   Wispots  Business Services
3          9-Aug-09  CollegeFoxesPackingBoxes     Lifestyle/Home
4          9-Aug-09                  IonicEar      Software/Tech

                           Business Description Pitchers Gender  …  \
0           Ava The Elephant - Baby and Child Care          Female  …
1           Mr. Tod's Pie Factory - Specialty Food            Male  …
2                      Wispots - Consumer Services            Male  …
3  College Foxes Packing Boxes - Consumer Services            Male  …
4                          Ionic Ear - Novelties            Male  …

  Kevin O Leary Investment Equity Guest Investment Amount  \
0                            NaN                      NaN
1                            NaN                      NaN
2                            NaN                      NaN
3                            NaN                      NaN
```

```
4                              NaN                    NaN

  Guest Investment Equity Guest Name Barbara Corcoran Present  \
0                     NaN        NaN                        1.0
1                     NaN        NaN                        1.0
2                     NaN        NaN                        1.0
3                     NaN        NaN                        1.0
4                     NaN        NaN                        1.0

   Mark Cuban Present  Lori Greiner Present  Robert Herjavec Present  \
0                 0.0                   0.0                      1.0
1                 0.0                   0.0                      1.0
2                 0.0                   0.0                      1.0
3                 0.0                   0.0                      1.0
4                 0.0                   0.0                      1.0

   Daymond John Present  Kevin O Leary Present
0                   1.0                    1.0
1                   1.0                    1.0
2                   1.0                    1.0
3                   1.0                    1.0
4                   1.0                    1.0

[5 rows x 50 columns]
```

[3]:
```python
explore_df.shape
```

[3]:
```
(1274, 50)
```

[4]:
```python
explore_df.isna().sum()
```

[4]:
```
Season Number                    0
Season Start                     0
Season End                       0
Episode Number                   0
Pitch Number                     0
Original Air Date                0
Startup Name                     0
Industry                         0
Business Description             0
Pitchers Gender                  7
Pitchers City                  772
Pitchers State                 528
Pitchers Average Age           936
Entrepreneur Names             495
Company Website                758
Multiple Entrepreneurs         427
```

```
US Viewership                             0
Original Ask Amount                       0
Original Offered Equity                   0
Valuation Requested                       0
Got Deal                                  0
Total Deal Amount                       509
Total Deal Equity                       509
Deal Valuation                          509
Number of sharks in deal                509
Investment Amount Per Shark             509
Equity Per Shark                        509
Royalty Deal                           1199
Loan                                   1222
Barbara Corcoran Investment Amount     1154
Barbara Corcoran Investment Equity     1154
Mark Cuban Investment Amount           1044
Mark Cuban Investment Equity           1044
Lori Greiner Investment Amount         1075
Lori Greiner Investment Equity         1075
Robert Herjavec Investment Amount      1153
Robert Herjavec Investment Equity      1153
Daymond John Investment Amount         1163
Daymond John Investment Equity         1163
Kevin O Leary Investment Amount        1157
Kevin O Leary Investment Equity        1157
Guest Investment Amount                1169
Guest Investment Equity                1169
Guest Name                             1169
Barbara Corcoran Present                376
Mark Cuban Present                      373
Lori Greiner Present                    373
Robert Herjavec Present                 377
Daymond John Present                    376
Kevin O Leary Present                   376
dtype: int64
```

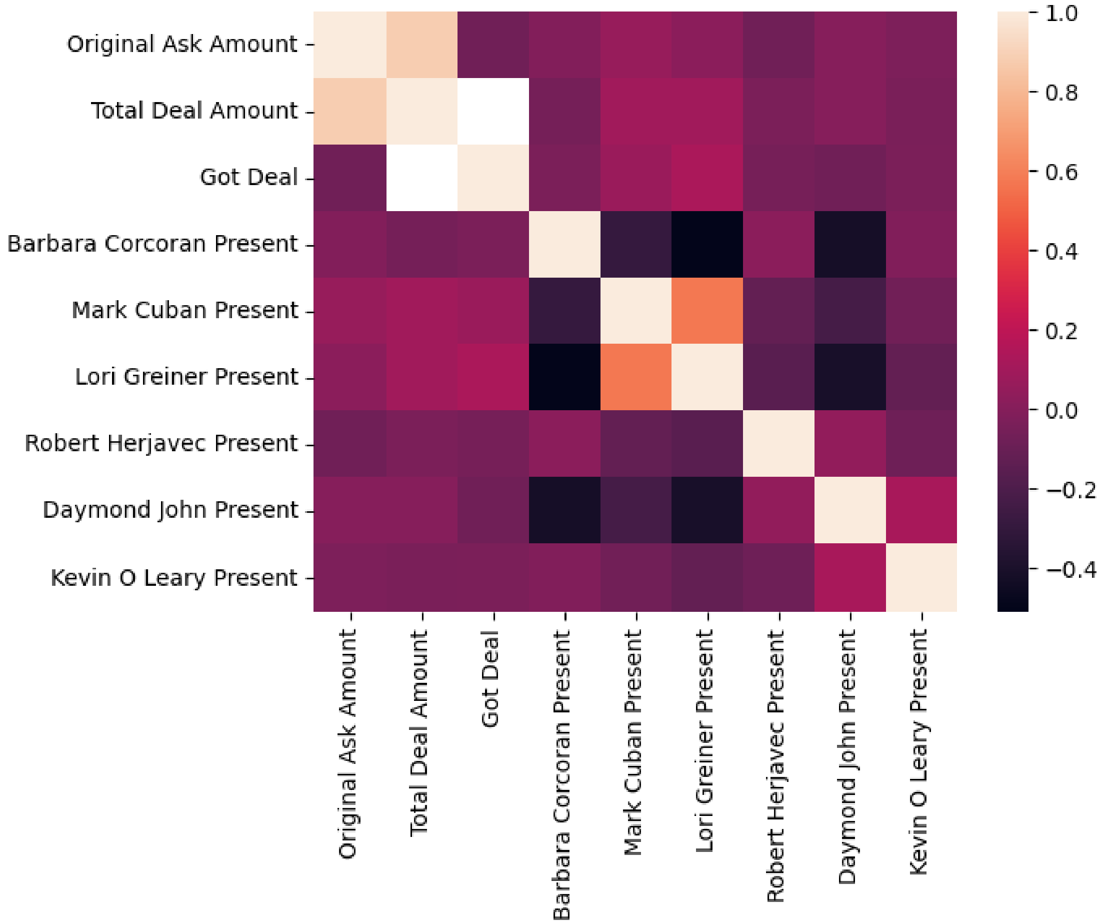
[5]:
```
kept_df = explore_df[['Industry','Pitchers Gender', 'Original Ask Amount',␣
↪'Total Deal Amount',
                      'Got Deal', 'Barbara Corcoran Present', 'Mark Cuban␣
↪Present', 'Lori Greiner Present',
                      'Robert Herjavec Present', 'Daymond John Present', 'Kevin␣
↪O Leary Present', 'Pitchers State']]

kept_df.shape
```

[5]:
```
(1274, 12)
```

[6]:
```
initial_drop = kept_df.dropna()
initial_drop.shape
```

[6]: (405, 12)

[7]:
```
# Calculating the correlations and then visualizing the relationships
corr_df = kept_df.corr()
sns.heatmap(corr_df)
plt.show()
```



"Total Deal Amount" has a perfect relationship with "Got Deal" and a very high correlation with "Original Ask Amount", "Total Deal Amount" will not be used of the model.

[8]:
```
# Dropping high correlation feature
v2_kept_df = kept_df.drop('Total Deal Amount', axis=1)
v2_kept_df.shape
```

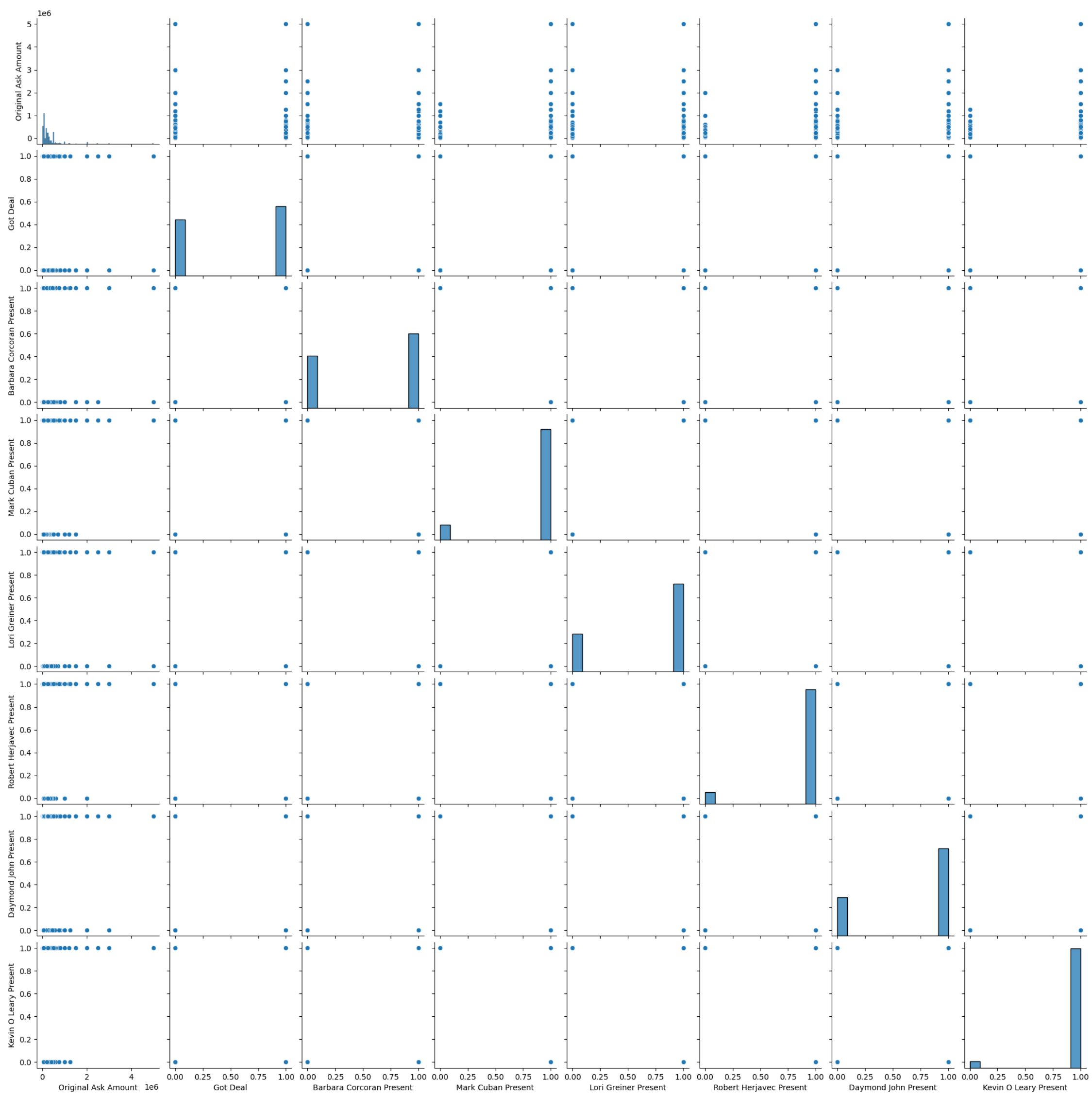
[8]: (1274, 11)

[9]:
```
dropping_nulls_df = v2_kept_df.dropna()
dropping_nulls_df.shape
```

[9]: (734, 11)

2 Visulaizing the data

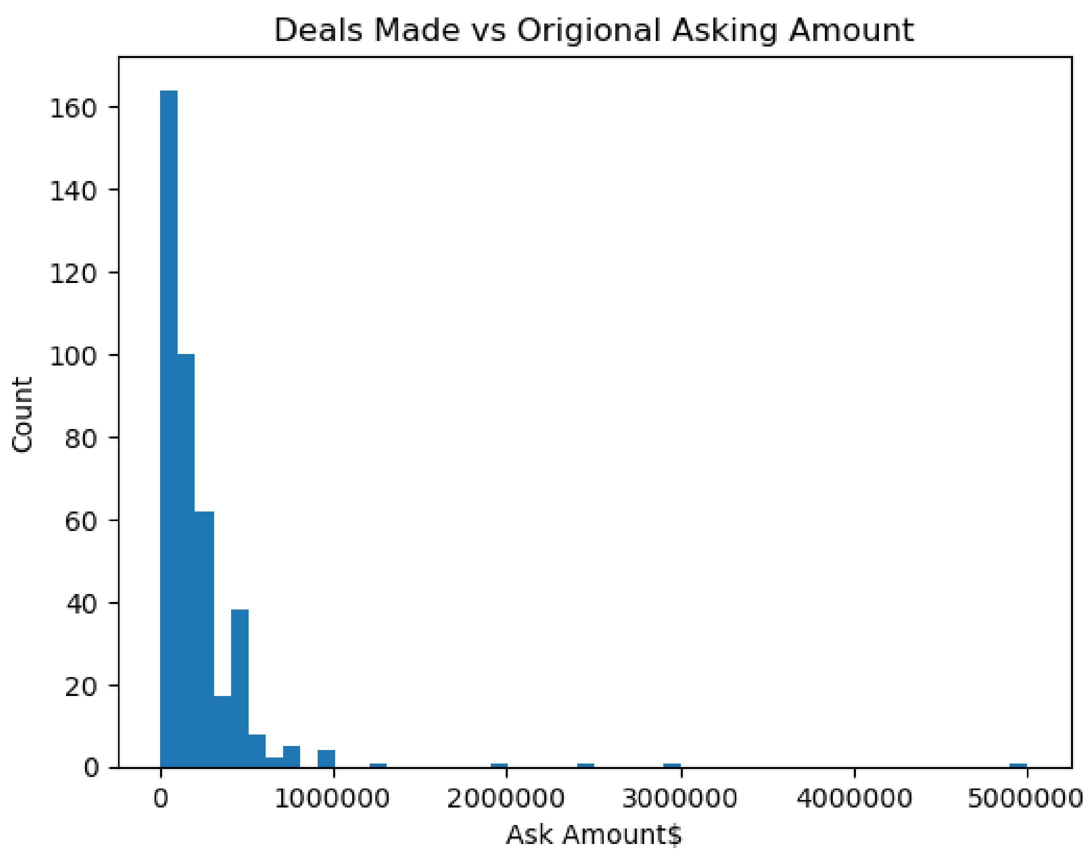
[10]:
```
sns.pairplot(dropping_nulls_df)
plt.show()
```



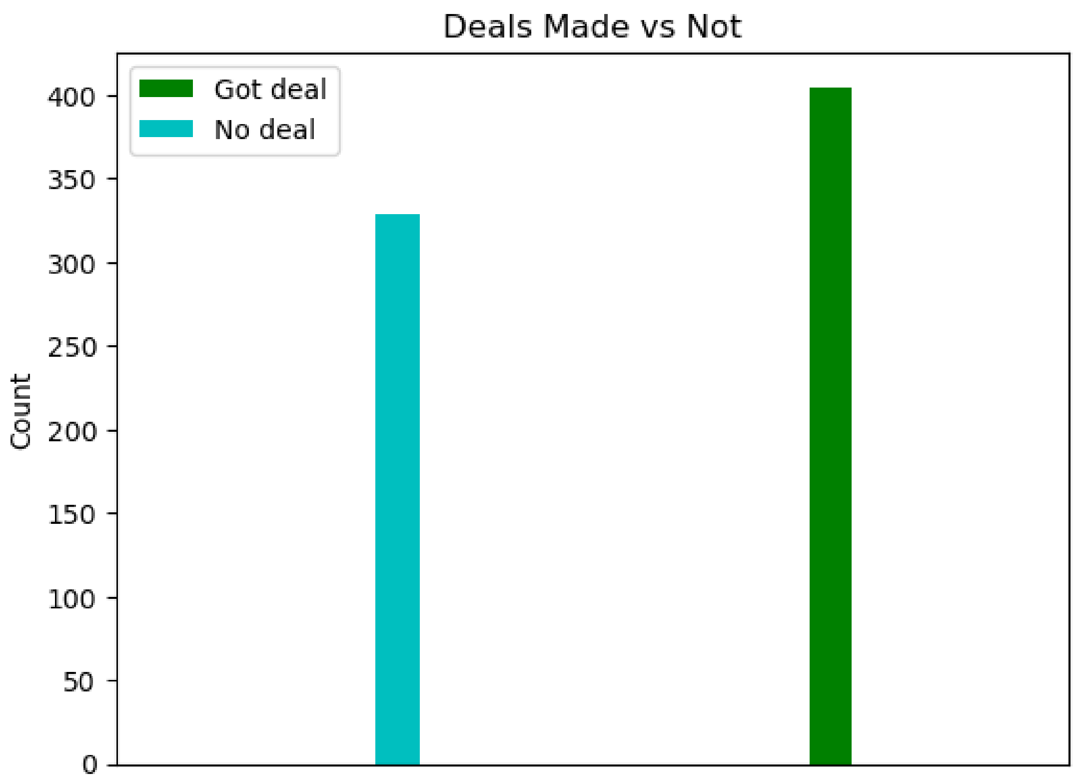
[11]:
```
# Creating a dataset of only companies that did and did not get deals
yes_deal_df = dropping_nulls_df[dropping_nulls_df['Got Deal']==1]
```

```
no_deal_df = dropping_nulls_df[dropping_nulls_df['Got Deal']!=1]
```

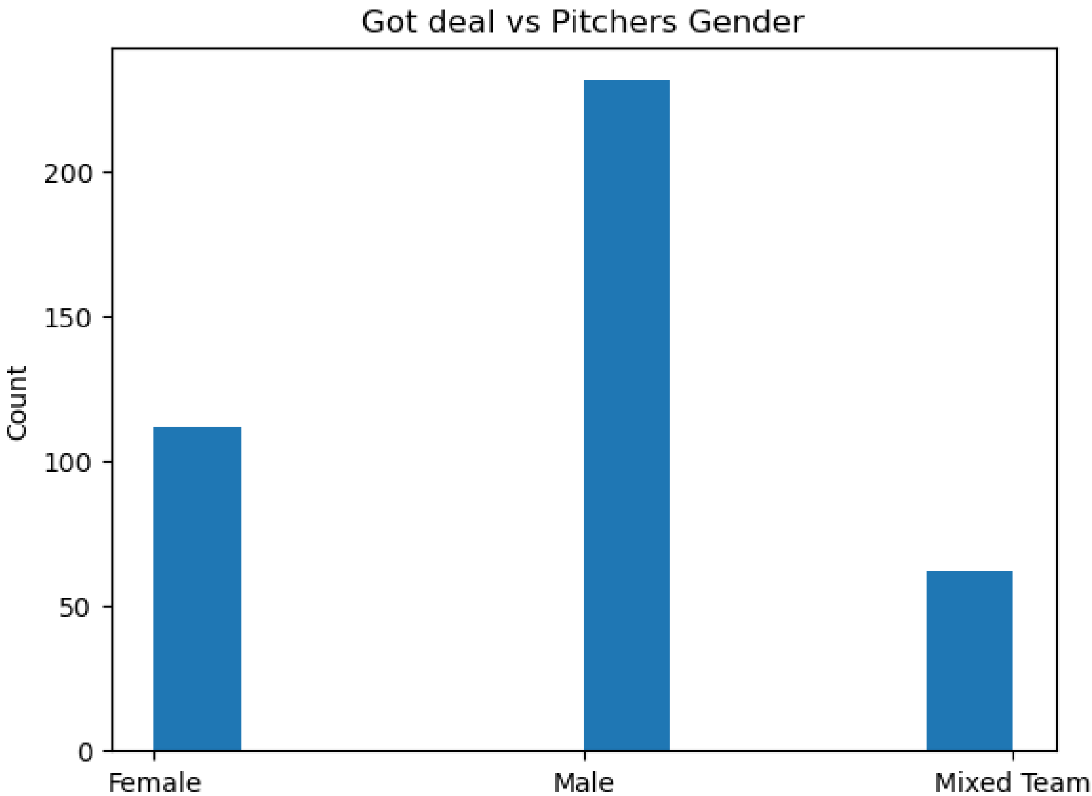
[12]:
```
# Histogram of companies that did secure a deal and how much they asked for
fig, ax=plt.subplots()
# Using the below code to remove scientific notation from the x-axis
ax.ticklabel_format(style='plain')
ax.hist(yes_deal_df['Original Ask Amount'], bins=(50))
ax.set_title('Deals Made vs Origional Asking Amount')
ax.set_xlabel('Ask Amount$')
ax.set_ylabel('Count')
plt.show()
```



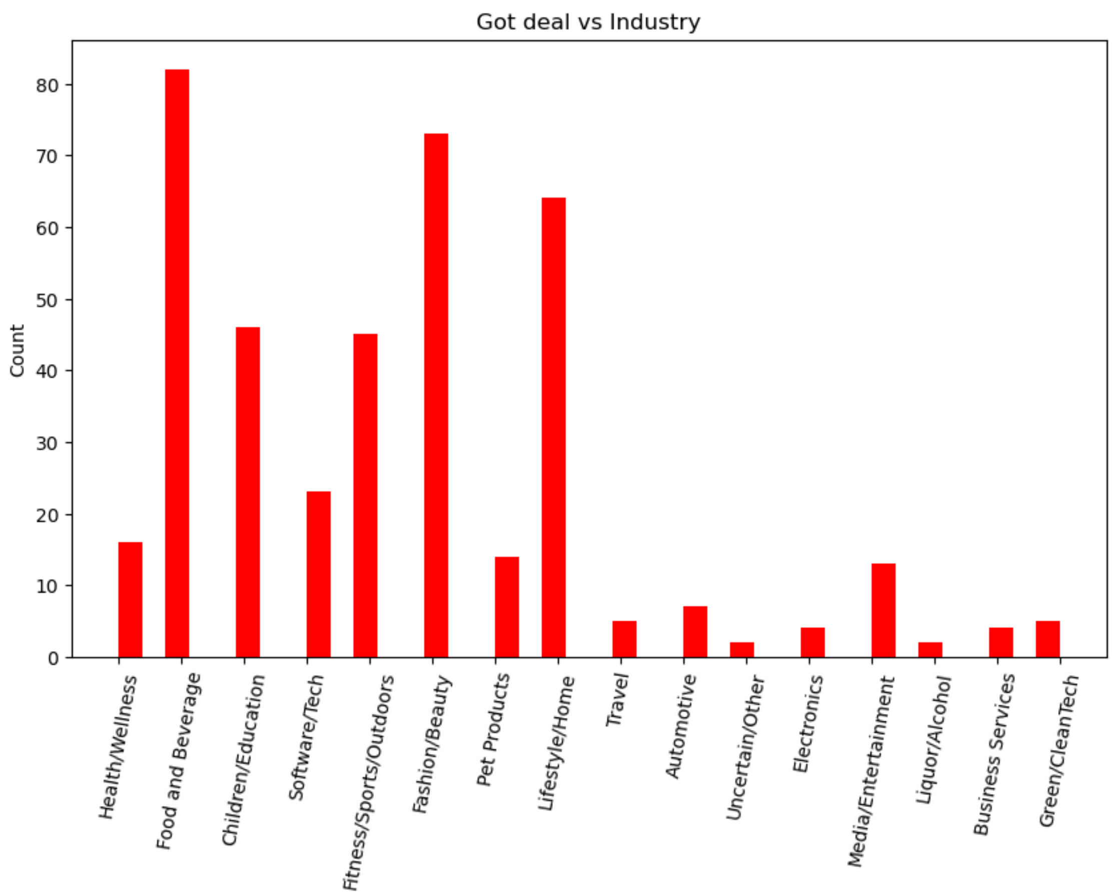
[13]:
```
# Histogram of companies that did and did not secure deals
plt.hist(yes_deal_df['Got Deal'], color='g', label='Got deal')
plt.hist(no_deal_df['Got Deal'], color='c', label='No deal')
plt.legend()
plt.title('Deals Made vs Not')
plt.xticks([])
plt.ylabel('Count')
plt.show()
```



[14]:
```
# Histogram of companies that secured deals by gender
plt.hist(yes_deal_df['Pitchers Gender'])
plt.title('Got deal vs Pitchers Gender')
plt.ylabel('Count')
plt.show()
```



[15]:
```
# Histogram of deals made by industry
plt.figure(figsize=(10,6))
plt.hist(yes_deal_df['Industry'], bins=40, color='r')
plt.xticks(rotation=80)
plt.title('Got deal vs Industry')
plt.ylabel('Count')
plt.show()
```



3 Creating Models

3.0.1 Logistic regression

[16]:
```
# Loading in the necessary packages
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score,␣
 ↪recall_score, ConfusionMatrixDisplay
```

[17]:
```
# Creating a feature and target dataframe
target_df = dropping_nulls_df['Got Deal']
feature_df = dropping_nulls_df.drop('Got Deal', axis=1)

# Creating a dummy variable dataframe to handle non-numeric features
dummy_df = pd.get_dummies(feature_df)
```

```
print(target_df.shape)
print(dummy_df.shape)
```

```
(734,)
(734, 72)
```

[18]:
```
# Splitting the data
df_train, df_test, target_train, target_test = train_test_split(dummy_df,␣
↪target_df, test_size=30, random_state=200)
print(df_train.shape, df_test.shape, target_train.shape, target_test.shape)

# Scaling the data to cover the wide range of feature values
scaler = StandardScaler()
df_train_scaled = scaler.fit_transform(df_train)
df_test_scaled = scaler.transform(df_test)
```

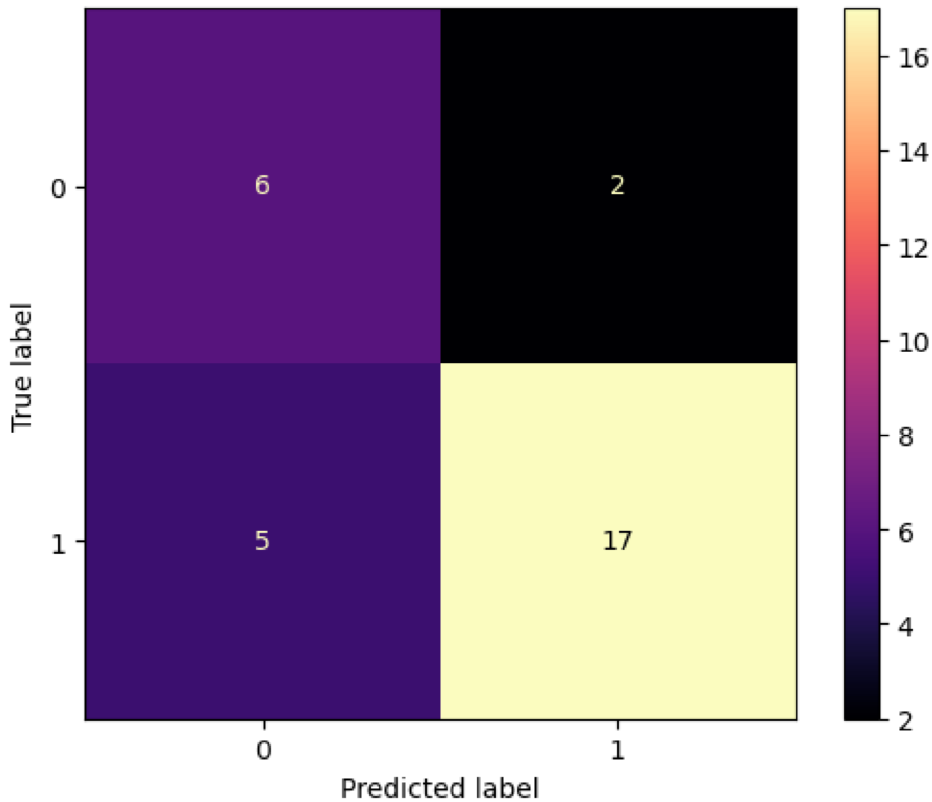
```
(704, 72) (30, 72) (704,) (30,)
```

[19]:
```
# Creating an instance of the regressor
log_reg = LogisticRegression(random_state=20).fit(df_train_scaled, target_train)

# Making a prediction of the y
predicted_test = log_reg.predict(df_test_scaled)
```

[33]:
```
# Calculating metrics
cm = confusion_matrix(predicted_test, target_test)
accuracy = round(accuracy_score(predicted_test, target_test), 2)*100
precison = round(precision_score(predicted_test, target_test), 2)*100
recall = round(recall_score(predicted_test, target_test), 2)*100

print(f'The accuracy is: {accuracy}%\nThe precision is: {precison}%\nThe recall␣
↪is: {recall}%')
```

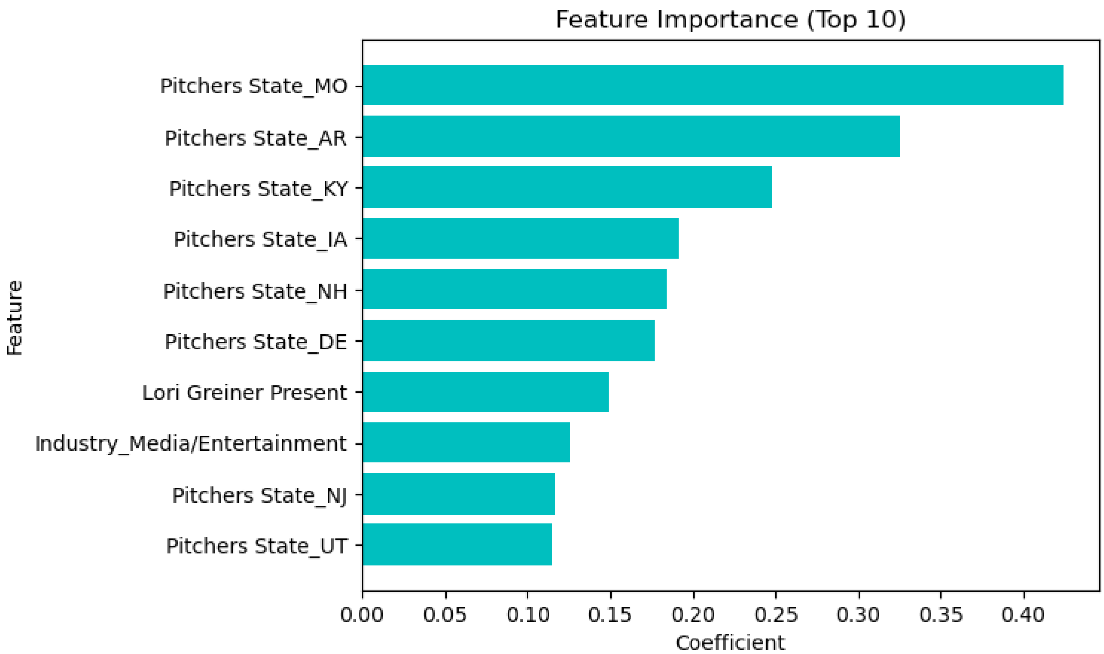
```
The accuracy is: 77.0%
The precision is: 89.0%
The recall is: 77.0%
```

[34]:
```
# Confusion matrix visualization
ConfusionMatrixDisplay(cm).plot(cmap='magma')
plt.show()
```



[22]:
```
# Determining feature importance
## Setting the coeffienet values and feature names to separate variables
coefficients = log_reg.coef_[0]
features = df_train.columns

# Creating a dataframe for each feature and the respective f
coef_df = pd.DataFrame(data=coefficients, index=features,␣
↪columns=['Coefficients']).sort_values(by=['Coefficients'],
                                                                        ␣
↪                    ascending=False)
```

[23]:
```
# Plotting the feature importance
plt.barh(coef_df.index[:10], coef_df['Coefficients'].head(10), color='c')
plt.gca().invert_yaxis()
plt.title('Feature Importance (Top 10)')
plt.xlabel('Coefficient')
plt.ylabel('Feature')
plt.show()
```



3.0.2 State related features appear to dominate so rerunning the model without state feature

[24]:
```
# Creating a new dataset without pitcher state
n_s_df = explore_df[['Industry','Pitchers Gender', 'Original Ask Amount',
                     'Got Deal', 'Barbara Corcoran Present', 'Mark Cuban␣
↪Present', 'Lori Greiner Present',
                     'Robert Herjavec Present', 'Daymond John Present', 'Kevin␣
↪O Leary Present']]

# Dropping nulls
n_s_dropping_nulls_df = n_s_df.dropna()
n_s_dropping_nulls_df.shape
```

[24]:
```
(895, 10)
```

[25]:
```
# Creating a feature and target df
n_s_target_df = n_s_dropping_nulls_df['Got Deal']
n_s_feature_df = n_s_dropping_nulls_df.drop('Got Deal', axis=1)

# Creating a dummy variable dataframe
n_s_dummy_df = pd.get_dummies(n_s_feature_df)

print(n_s_target_df.shape)
print(n_s_dummy_df.shape)
```

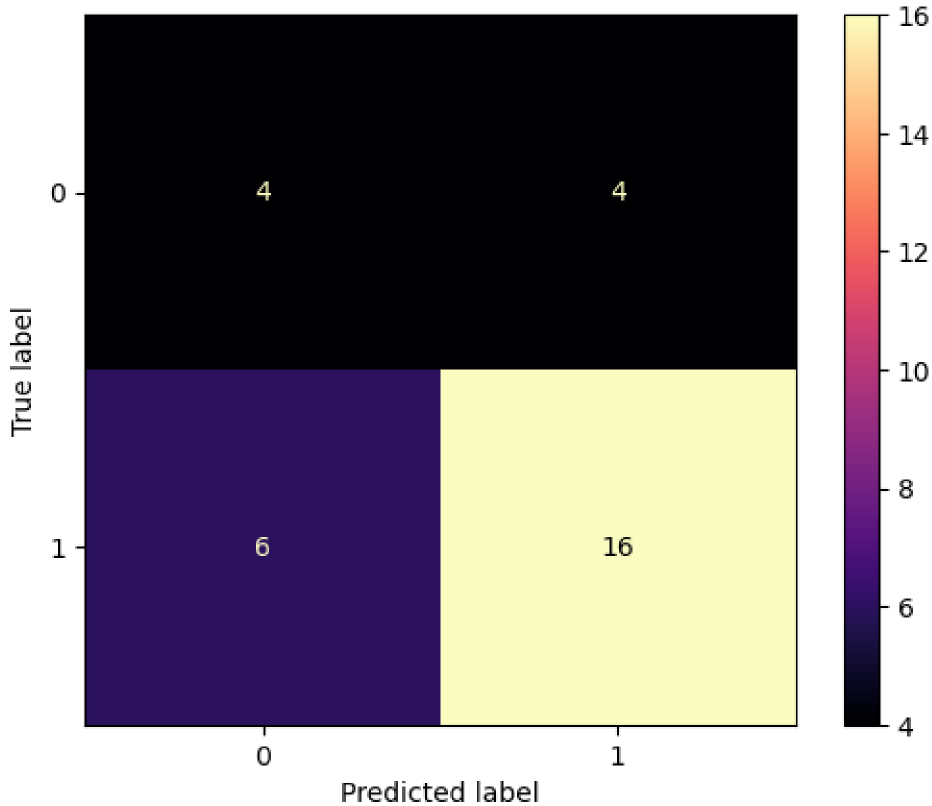
```
(895,)
(895, 26)
```

[26]:
```
# Splitting the data
n_s_train, n_s_test, n_s_target_train, n_s_target_test =␣
 ↪train_test_split(n_s_dummy_df, n_s_target_df, test_size=30,
                                                                                ␣
 ↪random_state=200)

# Scaling the data
no_state_scaler = StandardScaler()
n_s_df_train_scaled = no_state_scaler.fit_transform(n_s_train)
n_s_df_test_scaled = no_state_scaler.transform(n_s_test)

# Creating an instance
n_s_log_reg = LogisticRegression(random_state=20).fit(n_s_df_train_scaled,␣
 ↪n_s_target_train)
n_s_predicted_test = n_s_log_reg.predict(n_s_df_test_scaled)
```

[27]:
```
# Calculating metrics
n_s_cm = confusion_matrix(n_s_predicted_test, n_s_target_test)
n_s_accuracy = round(accuracy_score(n_s_predicted_test, n_s_target_test), 2)*100
n_s_precison = round(precision_score(n_s_predicted_test, n_s_target_test),␣
 ↪2)*100
n_s_recall = round(recall_score(n_s_predicted_test, n_s_target_test), 2)*100

print(f'The no state accuracy is: {n_s_accuracy}\nThe no state precision is:␣
 ↪{n_s_precison}\nThe no state recall is: {n_s_recall}')
```

```
The no state accuracy is: 67.0
The no state precision is: 80.0
The no state recall is: 73.0
```

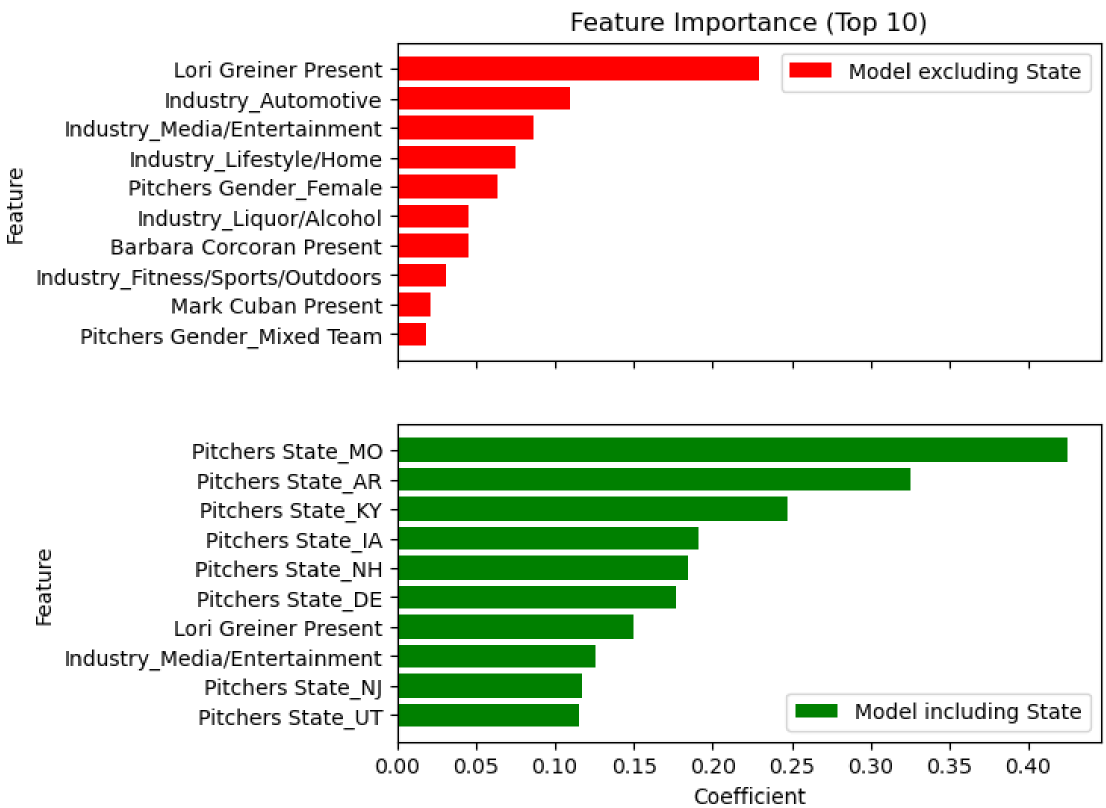
[35]:
```
# Confusion matrix visualization
ConfusionMatrixDisplay(n_s_cm).plot( cmap='magma')
plt.show()
```



[29]:

```
# Determining feature importance
## Setting the coeffienet values and feature names to separate variables
n_s_coefficients = n_s_log_reg.coef_[0]
n_s_features = n_s_train.columns

# Creating a dataframe for each feature and the respective f
n_s_coef_df = pd.DataFrame(data=n_s_coefficients, index=n_s_features,
                           columns=['Coefficients']).
 ↪sort_values(by=['Coefficients'], ascending=False)

# Plotting the feature importance
fig, ax = plt.subplots(2, figsize=(6,6), sharex=True)
ax[0].barh(n_s_coef_df.index[:10], n_s_coef_df['Coefficients'].head(10),␣
 ↪label=('Model excluding State'), color='r')
ax[0].invert_yaxis()
ax[0].legend()
ax[0].set_title('Feature Importance (Top 10)')
ax[0].set_ylabel('Feature')
```

```
ax[1].barh(coef_df.index[:10], coef_df['Coefficients'].head(10), label=('Model␣
↪including State'), color='g')
ax[1].invert_yaxis()
ax[1].set_xlabel('Coefficient')
ax[1].set_ylabel('Feature')
ax[1].legend()
plt.show()
```



4 Metric Recap

[36]:
```
print(f'Confusion Matrix\n{cm}\n')
print(f'The accuracy is: {accuracy}%\nThe precision is: {precison}%\nThe recall␣
↪is: {recall}%\n','-'*30,'\n','-'*30)

print(f'No State Confusion Matrix\n{n_s_cm}\n')
print(f'The no state accuracy is: {n_s_accuracy}%\nThe no state precision is:␣
↪{n_s_precison}%\nThe no state recall is: {n_s_recall}%')
```

```
Confusion Matrix
[[ 6  2]
 [ 5 17]]
```

```
The accuracy is: 77.0%
The precision is: 89.0%
The recall is: 77.0%
 ------------------------------
 ------------------------------
No State Confusion Matrix
[[ 4  4]
 [ 6 16]]

The no state accuracy is: 67.0%
The no state precision is: 80.0%
The no state recall is: 73.0%
```

5 Conclusion

The original logistic regression model produced respectable metrics accross the board thus making the model useful for future use. Potential concerns are when it comes to the pitches states, high outcomes for respective states could be due to filming primarily being in the states thus limiting the ease of access for others. Additionally the final data size was 734 rows and 11 features. I would of liked to of had a minimum of 1000 rows; however, 734:11 is a decent ratio for modeling.