Predictive Analytics: Assignment 3.2

Joshua Greenert

DSC630-T301 Predictive Analytics

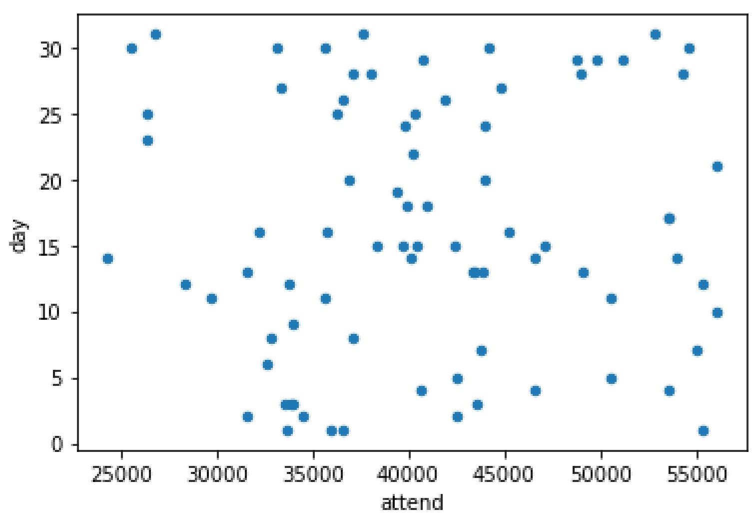
12/14/2022

In [1]:
```
# Import the required libaries.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Pull in the data to begin preparation.
df_mlb = pd.read_csv('dodgers-2022.csv')
df_mlb.head(5)
```

Out[1]:

	month	day	attend	day_of_week	opponent	temp	skies	day_night	cap	shirt	fireworks	bobblehead
0	APR	10	56000	Tuesday	Pirates	67	Clear	Day	NO	NO	NO	NO
1	APR	11	29729	Wednesday	Pirates	58	Cloudy	Night	NO	NO	NO	NO
2	APR	12	28328	Thursday	Pirates	57	Cloudy	Night	NO	NO	NO	NO
3	APR	13	31601	Friday	Padres	54	Cloudy	Night	NO	NO	YES	NO
4	APR	14	46549	Saturday	Padres	57	Cloudy	Night	NO	NO	NO	NO

In [2]:
```
# Create some scatter plots to check the data.
ax = df_mlb.plot.scatter(x='attend', y='day', colormap='viridis')
```



There's nothing statistically significant with the day from this observation. The hope was to find out whether there were more attendees during the beginning or end of the month. However, this appears to be scattered randomly.

Data Preparation

In [3]:
```
# Make dummies from the data.
df_mlb_dummies = pd.get_dummies(df_mlb)
```

```
# Show the new dataframe.
df_mlb_dummies.head(5)
```

Out[3]:

	day	attend	temp	month_APR	month_AUG	month_JUL	month_JUN	month_MAY	month_OCT	month_SEP
0	10	56000	67	1	0	0	0	0	0	0
1	11	29729	58	1	0	0	0	0	0	0
2	12	28328	57	1	0	0	0	0	0	0
3	13	31601	54	1	0	0	0	0	0	0
4	14	46549	57	1	0	0	0	0	0	0

5 rows × 46 columns

We can convert all of the categorical values to dummies so that the data can be researched further with predictions. Additionally, certain columns can be removed since they would provide potential for false positives. These columns include any of the columns that include a yes or no (cap, shirt, fireworks, bobblehead); the columns that are listed as no in this dataset provide no real insight to the outcome. Therefore, we can remove them to improve our future predictions/feature selection.

In [4]:

```
# Remove the columns that don't provide insight.
df_mlb_dummies.drop(['cap_NO', 'shirt_NO', 'fireworks_NO', 'bobblehead_NO'], axis=1, inp

# Show the dataframe.
df_mlb_dummies.head(5)
```

Out[4]:

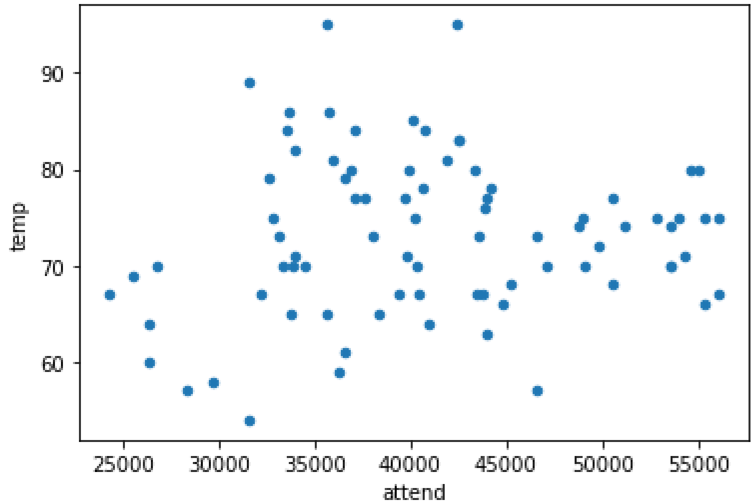
	day	attend	temp	month_APR	month_AUG	month_JUL	month_JUN	month_MAY	month_OCT	month_SEP
0	10	56000	67	1	0	0	0	0	0	0
1	11	29729	58	1	0	0	0	0	0	0
2	12	28328	57	1	0	0	0	0	0	0
3	13	31601	54	1	0	0	0	0	0	0
4	14	46549	57	1	0	0	0	0	0	0

5 rows × 42 columns

Charts and Graphs

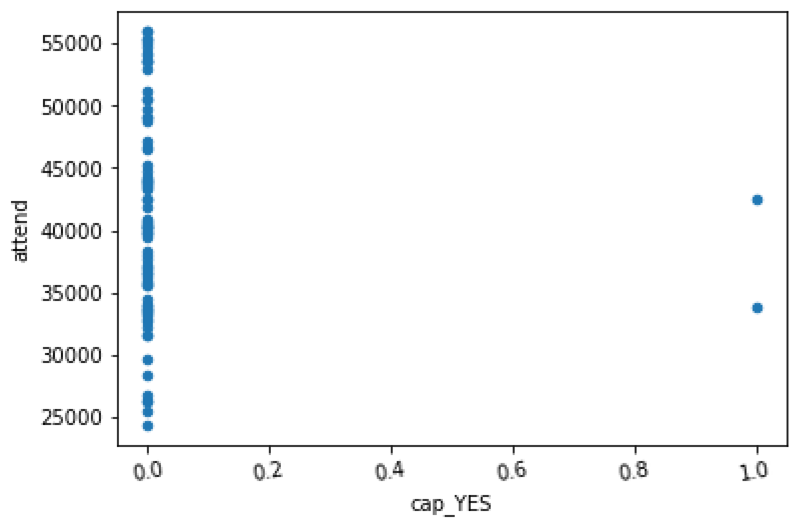
In [5]:

```
# Review the temperature.
ax = df_mlb_dummies.plot.scatter(x='attend', y='temp')
```

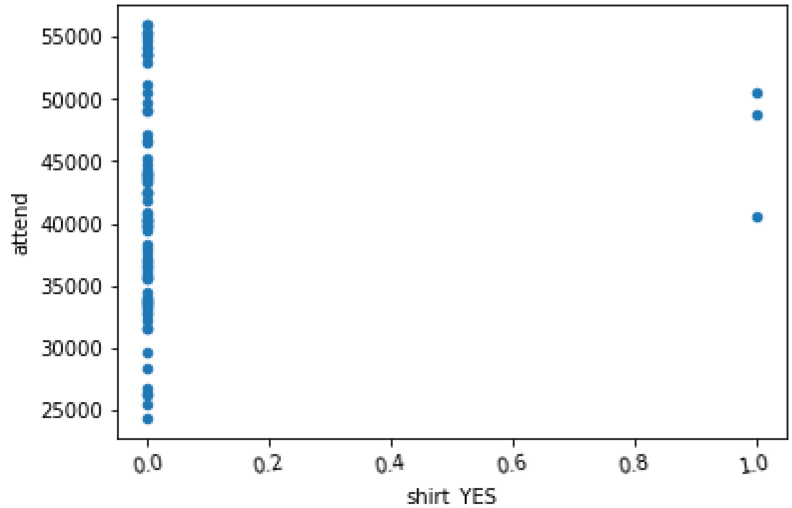


On the surface, this data doesn't look to provide much insight. However, the data shows that at the highest levels of attendance typically has a range of 68 to 80 degrees for the weather. This may prove useful later on the feature selection phase.

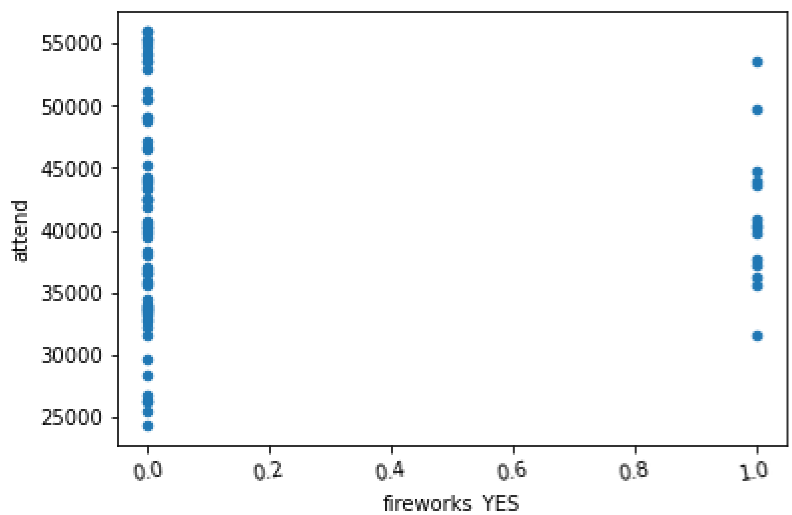
In [6]:
```
# Bar graphs with yes values for item giveaways.
# Set a new dataframe for this subplot process.
df_giveaways = pd.DataFrame(df_mlb_dummies, columns=['attend', 'cap_YES', 'shirt_YES', '

# Show the caps.
ax = df_giveaways.plot.scatter(x='cap_YES', y='attend', rot=10)
```



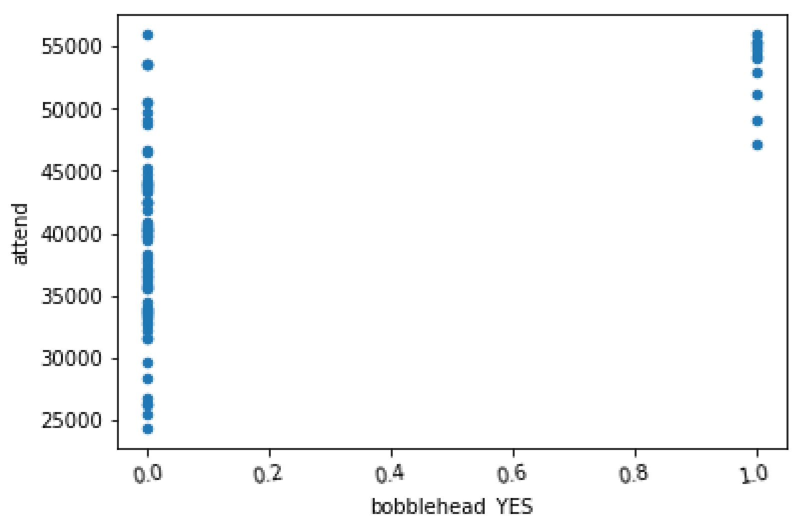
In [7]:
```
# Show the shirts.
ax = df_giveaways.plot.scatter(x='shirt_YES', y='attend', rot=10)
```



In [8]:
```
# Show the fireworks.
ax = df_giveaways.plot.scatter(x='fireworks_YES', y='attend', rot=10)
```



In [9]:
```
# Show the bobbleheads.
ax = df_giveaways.plot.scatter(x='bobblehead_YES', y='attend', rot=10)
```



When initially revewing the data, I presumed that the giveaways wouldn't be too insightful. After providing more scatter plots, it's clear that bobblehead giveaways appear during the same time as the highest attendance. Meanwhile, caps and fireworks didn't appear to be nearly as impactful. This is not conclusive yet, but may be a potential solution for the final recommendations.

Feature Selection

In [10]:
```
# Use a χ2-statistic selector to pick the five best features for this data (chi2)
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.model_selection import train_test_split

# Set the test and train data.
train, test = train_test_split(df_mlb_dummies)
```

In [11]:
```
# Set targets and features for training and test data.
target = train['attend']
features = train.drop(['attend'],axis=1).values

target_test = test['attend']
features_test = test.drop(['attend'],axis=1).values
```

In [12]:
```
# Select 3 features with the highest chi-squared statistics.
chi2_selector = SelectKBest(chi2, k = 3)
features_kbest = chi2_selector.fit_transform(features, target)
```

In [24]:
```
from numpy import array

# Set a new dataframe with the attend column removed.
df_dummies_no_attend = df_mlb_dummies.drop(['attend','day', 'temp', 'month_APR', 'month_

# Set the feature filter, set the feature names.
cols = chi2_selector.get_support(indices = True)
features_names = df_dummies_no_attend.columns[cols]

print(features_names)
```

```
Index(['month_MAY', 'month_OCT', 'fireworks_YES'], dtype='object')
```

The attend, day, and temp attributes were removed prior to feature selection to reduce features that are outside of our control. In fact, additional runs were prodcued that removed additional features that related to the weather since month_APR, month_AUG, and skies_Cloudy were the second set of results. After removing those additional fields, we can finally see some potential results. It would appear that fireworks being used has potential to bringing in more attendance. This is still inconclusive. After realizing that we have features listed that are outside of our control, we should reduce our features to gain more insights on pieces that we CAN control.

In [55]:
```
# Performing test again after reduction of features.
features_reduced = train.drop(['attend','day', 'temp', 'month_APR', 'month_AUG', 'skies_
                              'month_MAY', 'month_OCT', 'month_SEP', 'day_night_Day','d

# Select 3 features with the highest chi-squared statistics.
chi2_selector = SelectKBest(chi2, k = 3)
features_kbest = chi2_selector.fit_transform(features_reduced, target)

# Set the feature filter, set the feature names.
cols = chi2_selector.get_support(indices = True)
features_names = df_dummies_no_attend.columns[cols]

print(features_names)
```

```
Index(['opponent_Angels', 'opponent_Reds', 'opponent_Snakes'], dtype='object')
```

It appears as though our reduction in features has highlighed some intersting data now. We are consistently showing that certain opponents have the potential to yield larger attendance numbers. This makes sense when we consider that fans of other teams would be attending to support their team as opposed to our team. It's likely that we should get a higher turnout for certain opponents based on their amount of supporters within the area. However, I think we can take one more look at our giveaways and see if there is more to learn from those features as well.

In [38]:
```
# Performing test again after reduction of features.
features_reduced = train[['cap_YES', 'shirt_YES','fireworks_YES','bobblehead_YES']].valu

# Set new dataframe with only these features included.
df_dummmies_reduced = pd.DataFrame(df_dummies_no_attend, columns=['cap_YES', 'shirt_YES'

# Select 3 features with the highest chi-squared statistics.
chi2_selector = SelectKBest(chi2, k = 3)
features_kbest = chi2_selector.fit_transform(features_reduced, target)

# Set the feature filter, set the feature names.
cols = chi2_selector.get_support(indices = True)
features_names = df_dummmies_reduced.columns[cols]

print(features_names)
```

```
Index(['cap_YES', 'shirt_YES', 'fireworks_YES'], dtype='object')
```

Surprisingly, according to the chi2 selector, our best 3 features are predicted as the cap, shirt, then fireworks. Our findings earlier with the bobbleheads may have been misguided by the scatterplots.

Predictions and Metrics

In [54]:
```
# Run a linear regression and report the R2-value and RMSE on the train set.
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error

# Create the linear regression model
regression = LinearRegression()

# Fit the linear Regression
model = regression.fit(features, target)

# Calculate the values for train data.
target_probabilities = model.predict(features)

# Print the R2 and RMSE
print(f"R2: {regression.score(features, target)}")
print(f"RMSE: {mean_squared_error(target, target_probabilities, squared = False)}")
```

```
R2: 0.7366441310654005
RMSE: 4221.808226183758
```

Using our initial features with a linearRegession model, we can see scores of an RMSE of 4221 which indiate a poor fit for the data. However, the R2 value of 0.74 indicates a possibility of statistical significance; if all the features used together account for approximately 74% of the target, then we may be closer than we think.

In [56]:
```
from sklearn.tree import DecisionTreeClassifier

# Create the DecisionTreeClassifier model
decision_tree = DecisionTreeClassifier()

# Fit the DecisionTreeClassifier
model = decision_tree.fit(features_kbest, target)
```

```
# Calculate the values for train data.
target_probabilities = model.predict(features_kbest)

# Print the R2 and RMSE
print(f"R2: {decision_tree.score(features_kbest, target)}")
print(f"RMSE: {mean_squared_error(target, target_probabilities, squared = False)}")
```

```
R2: 0.08333333333333333
RMSE: 16192.402334119543
```

When we segment the data we thought to be significant (giveaways), we see the exact opposite. Our R2 value reduces to 0.08% which indicates that those features (cap, shirt, and fireworks) only account for 8 percent of the target. This appears to be highlighting that the giveaways are not worth the effort to gain additional fan attendance. Now we can try with features based only on the opponents to see if it is as significant as we thought.

In [73]:

```
# Create a list of the opponent features.
opponents = list(df_dummies_no_attend.columns)
list_opponents = []

# Use a for loop to remove all features not opponents.
for i in opponents:
    if i.find("opponent") != -1:
        list_opponents.append(i)

# Convert to list.
list_opponents = list(list_opponents)

# Create the train features with the new features.
features_opponents = pd.DataFrame(train, columns=list_opponents)

# Fit the DecisionTreeClassifier
model = decision_tree.fit(features_opponents, target)

# Calculate the values for train data.
target_probabilities = model.predict(features_opponents)

# Print the R2 and RMSE
print(f"R2: {decision_tree.score(features_opponents, target)}")
print(f"RMSE: {mean_squared_error(target, target_probabilities, squared = False)}")
```

```
R2: 0.26666666666666666
RMSE: 10695.80968261247
```

In [77]:

```
# Create a list of the opponent features.
columns = list(df_dummies_no_attend.columns)
list_days = []

# Use a for loop to remove all features not opponents.
for i in columns:
    if i.find("day_of_week") != -1:
        list_days.append(i)

# Convert to list.
list_days = list(list_days)

# Create the train features with the new features.
features_days = pd.DataFrame(train, columns=list_days)

# Fit the DecisionTreeClassifier
model = decision_tree.fit(features_days, target)

# Calculate the values for train data.
```

```
target_probabilities = model.predict(features_days)

# Print the R2 and RMSE
print(f"R2: {decision_tree.score(features_days, target)}")
print(f"RMSE: {mean_squared_error(target, target_probabilities, squared = False)}")
```

```
R2: 0.1333333333333333
RMSE: 12387.685190004897
```

From the two predictions above, it appears that the opponents are fairly significant in regards of predicting attendance. Additionally, it is likely that a certain day may average more attendance than others as well. We can perform the average of these days below to gain more insight.

In [103…:

```
# Group by the day of the week to get the average.
df_mlb.groupby(['day_of_week']).mean().sort_values(by='attend',ascending=False).head(10)
```

Out[103]:

day_of_week	day	attend	temp
Tuesday	14.076923	47741.230769	72.769231
Saturday	17.923077	43072.923077	72.692308
Sunday	16.615385	42268.846154	78.153846
Thursday	22.800000	40407.400000	72.400000
Friday	19.307692	40116.923077	69.692308
Wednesday	12.416667	37585.166667	73.000000
Monday	13.416667	34965.666667	72.833333

In [91]:

```
# Group by the opponent to get the average.
df_mlb.groupby(['opponent']).mean().sort_values(by='attend',ascending=False).head(10)
```

Out[91]:

opponent	day	attend	temp
Angels	12.000000	49777.333333	67.000000
Mets	22.000000	49586.250000	75.000000
Nationals	28.000000	49267.333333	70.333333
White Sox	16.000000	46382.000000	69.666667
Cubs	4.000000	44206.666667	76.333333
Padres	10.666667	42092.222222	71.444444
Phillies	17.000000	41897.000000	72.333333
Cardinals	16.428571	40853.285714	79.142857
Marlins	25.000000	40665.333333	74.000000
Reds	3.000000	40649.000000	70.000000

Based on the averaged data above, we can see that Tuesdays are the highest days for attendance; however, this day is closely followed by Saturday and Sunday leaving Monday as the lowest average day for attendance. Additionally, we can see from the groupby performed on the opponents that the Angels, Mets, Nationals, White sox, and Cubs show the most attendance on average. The Angels were predicted as one of the main best features in previous feature selections, but the Reds and Snakes were the next best features according to the prediction.

In [104…

```
# Group by the day of the week to get the average.
df_mlb.groupby(['bobblehead']).mean().sort_values(by='attend',ascending=False).head(2)
```

Out[104]:

	day	attend	temp
bobblehead			
YES	19.636364	53144.636364	74.181818
NO	15.585714	39137.928571	72.985714

After checking through each iteration of attendance with each specific giveaway, only Bobbleheads has a clear difference. It's likely that a mixture of teams and this particular giveaway may be significant.

Conclusion Summary

Based on the findings reported above, it is strongly likely that the highest number of turnout happens when certain opponents are versing our team. Specifically, we see a notable difference in turnout when the opposing teams Angels, Mets, Nationals, White Sox, and Cubs are versing our team. Likewise, we also see a much larger number of attendees on days where the bobblehead giveaways take place. To improve attendance for our team, we should be providing more bobbleheads as giveaways on days where the aforementioned teams are set to play.