Week 3 Homework - Jon Workman

In [69]:
```
from datetime import date
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.dummy import DummyClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
import seaborn as sns
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, accuracy_score
%matplotlib inline
```

Load dataset

In [70]:
```
# Load Seattle weather dataset
df = pd.read_csv('seattleWeather_1948-2017.csv').dropna()
```

Clean data

In [71]:
```
# Clean data
df.RAIN = df.RAIN.astype(int)
df['DATE'] =  pd.to_datetime(df['DATE'], format='%Y-%m-%d')
```

Create additional feature

In [72]:
```
# Average precipitation for last 2 days feature.
# Initially tried 7 days and 5 days. 2 days was the best predictor of raining on a given day.

df['average_PRCP'] = df['PRCP'].rolling(2).mean().round(2)
```

In [73]:
```
# Find any null values
df[df.isnull().any(axis=1)]
```

Out[73]:

	DATE	PRCP	TMAX	TMIN	RAIN	average_PRCP
0	1948-01-01	0.47	51	42	1	NaN

In [74]:
```
# Remove null values
df.drop(df.index[0], inplace=True)
```

In [75]:
```
df.head(5)
```

Out[75]:

	DATE	PRCP	TMAX	TMIN	RAIN	average_PRCP
1	1948-01-02	0.59	45	36	1	0.53
2	1948-01-03	0.42	45	35	1	0.50
3	1948-01-04	0.31	45	34	1	0.37
4	1948-01-05	0.17	45	32	1	0.24
5	1948-01-06	0.44	48	39	1	0.31

Split dataset into train and test

In [76]:
```
# Split dataset into train and test
start1 = pd.datetime(1950, 1, 1)
end1 = pd.datetime(2009, 12, 31)
start2 = pd.datetime(2010,1,1)
end2 = pd.datetime(2017,12,31)

df_train = df[(df.DATE >= start1) & (df.DATE <= end1)]
df_test = df[(df.DATE >= start2) & (df.DATE <= end2)]
```

In [77]:
```
# Drop all features excluding TMAX and average_PRCP. Can't use PRCP to predict future rain day
s.
feature_cols = ['TMAX','average_PRCP']

X = df_train[feature_cols]
y = df_train['RAIN']
X_test = df_test[feature_cols]
y_test = df_test['RAIN']
```

Null accuracy

In [78]:
```
# calculate null accuracy
# Future models should be better than this, otherwise don't waste your time.
dumb = DummyClassifier(strategy='most_frequent')
dumb.fit(X, y)
y_dumb = dumb.predict(X_test)
print('Null accuracy:', metrics.accuracy_score(y_test, y_dumb))
```

```
Null accuracy: 0.552495697074
```

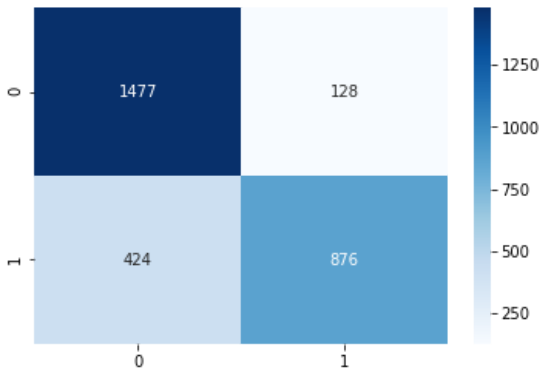
Logistic regression model

In [104]:
```
logreg = LogisticRegression()
logreg.fit(X, y)
y_pred = logreg.predict(X_test)
print ('Accuracy:{:.2}'.format(metrics.accuracy_score(y_test, y_pred)))
```

```
Accuracy:0.81
```

In [80]:
```
logmodel = LogisticRegression()
logmodel.fit(X,y)
y_pred = logmodel.predict(X_test)
sns.heatmap(confusion_matrix(y_test, y_pred), cmap='Blues', annot=True, fmt='g')
```

Out[80]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1e655828>
```



In [81]:
```
print(classification_report(y_test, y_pred))
```

```
             precision    recall  f1-score   support

          0       0.78      0.92      0.84      1605
          1       0.87      0.67      0.76      1300

avg / total       0.82      0.81      0.81      2905
```

In [107]:
```
# Predicting using train set
y_pred = logmodel.predict(X)
print('Accuracy: {:.2}'.format(metrics.accuracy_score(y,y_pred)))
```

```
Accuracy: 0.81
```

In [108]:
```
# Predicting using test set
y_pred = logmodel.predict(X_test)
print('Accuracy: {:.2}'.format(metrics.accuracy_score(y_test,y_pred)))
```

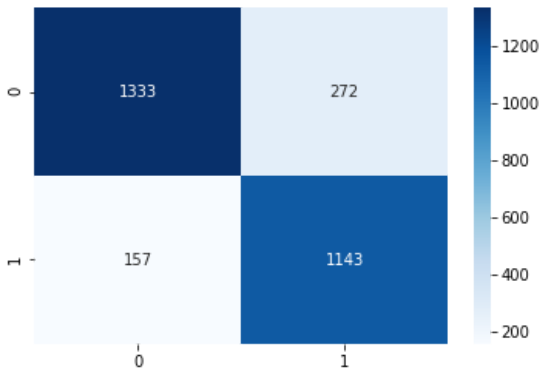
```
Accuracy: 0.81
```

K-Nearest Neighbour model

In [97]:
```
knn = KNeighborsClassifier(n_neighbors=11)
knn.fit(X,y)
y_pred = knn.predict(X_test)
print('Accuracy:{:.2}'.format(metrics.accuracy_score(y_test, y_pred)))
sns.heatmap(confusion_matrix(y_test, y_pred), cmap='Blues', annot=True, fmt='g')
```

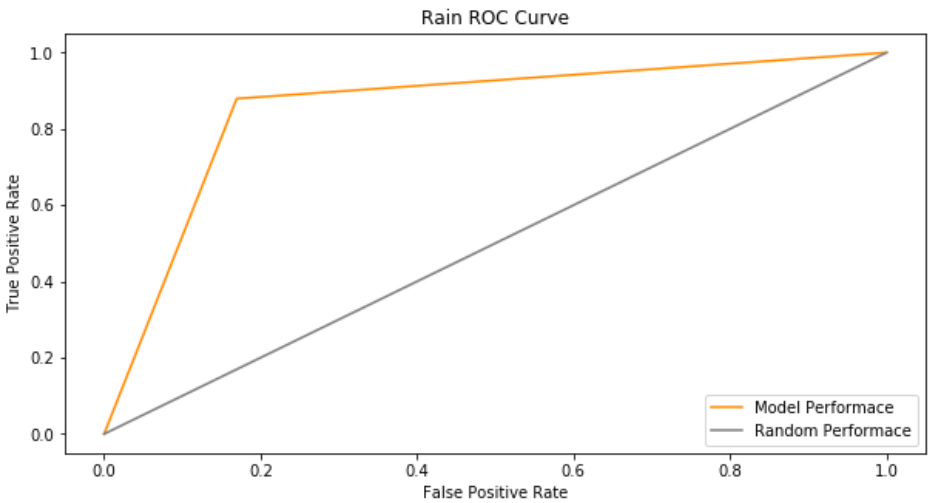
```
Accuracy:0.85
```

Out[97]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1eb10a58>
```



In [98]:
```
fig, ax = plt.subplots(1, figsize=(10, 5))
plt.plot(fpr, tpr, color='darkorange', label='Model Performace')
plt.plot([0, 1], [0, 1], color='gray', label='Random Performace')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Rain ROC Curve')
plt.legend(loc="lower right")
```

Out[98]:
```
<matplotlib.legend.Legend at 0x1a1ec551d0>
```



In [99]:
```
print(classification_report(y_test,y_pred))
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
```

```
             precision    recall  f1-score   support

          0       0.89      0.83      0.86      1605
          1       0.81      0.88      0.84      1300

avg / total       0.86      0.85      0.85      2905
```

In [109]:
```
# Predicting using train set
y_pred = knn.predict(X)
print('Accuracy: {:.2}'.format(metrics.accuracy_score(y,y_pred)))
```

```
Accuracy: 0.86
```

In [111]:
```
# Predicting using test set
y_pred = knn.predict(X_test)
print('Accuracy: {:.2}'.format(metrics.accuracy_score(y_test,y_pred)))
```

```
Accuracy: 0.85
```