# Pima_Native_Americans_2

March 5, 2020

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import scale
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_squared_error as MSE
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier as KNN
from sklearn.ensemble import VotingClassifier
```

In [2]:
```
df = pd.read_csv("C:\\Users\\User\\Documents\\Edx_project\\Pima Indians.csv")
```

In [3]:
```
df.describe().transpose()
```

Out[3]:

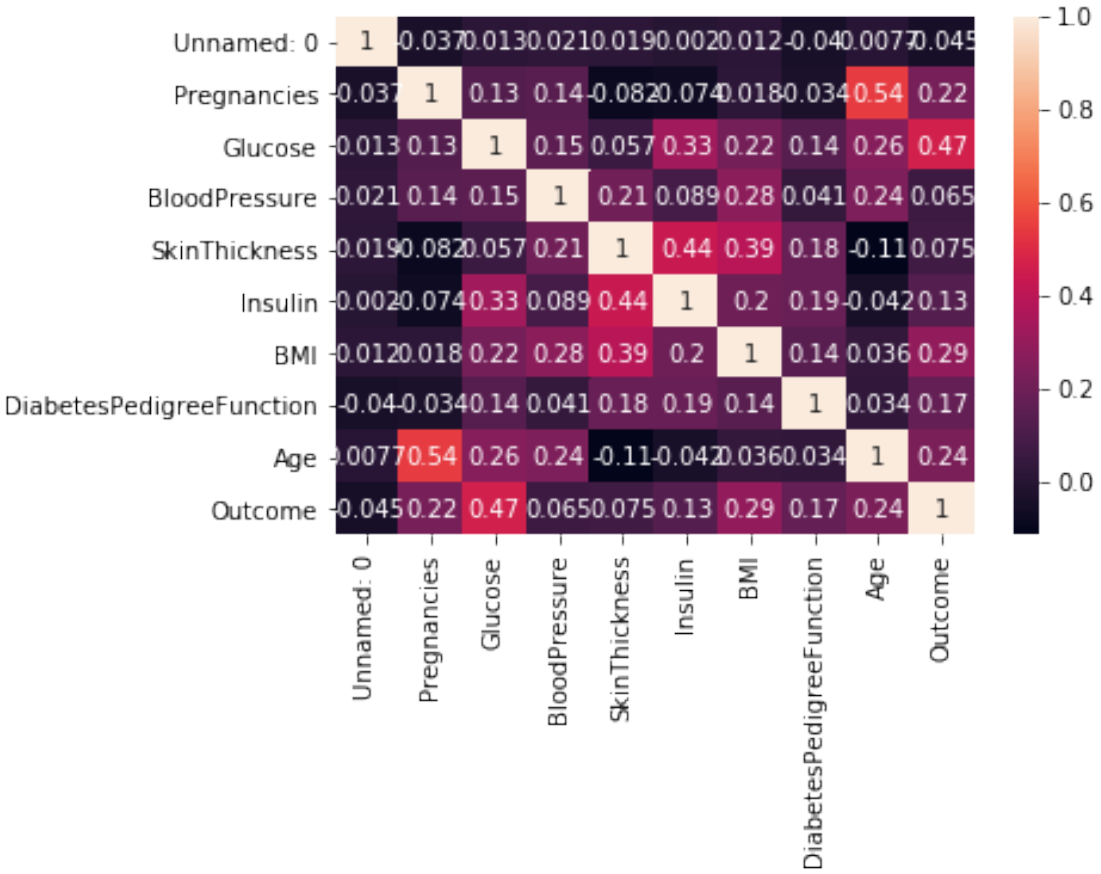
| | count | mean | std | min | 25% | \ |
|---|---|---|---|---|---|---|
| Unnamed: 0 | 768.0 | 384.500000 | 221.846794 | 1.000 | 192.75000 | |
| Pregnancies | 768.0 | 3.845052 | 3.369578 | 0.000 | 1.00000 | |
| Glucose | 768.0 | 120.894531 | 31.972618 | 0.000 | 99.00000 | |
| BloodPressure | 768.0 | 69.105469 | 19.355807 | 0.000 | 62.00000 | |
| SkinThickness | 768.0 | 20.536458 | 15.952218 | 0.000 | 0.00000 | |
| Insulin | 768.0 | 79.799479 | 115.244002 | 0.000 | 0.00000 | |
| BMI | 768.0 | 31.992578 | 7.884160 | 0.000 | 27.30000 | |
| DiabetesPedigreeFunction | 768.0 | 0.471876 | 0.331329 | 0.078 | 0.24375 | |
| Age | 768.0 | 33.240885 | 11.760232 | 21.000 | 24.00000 | |
| Outcome | 768.0 | 0.348958 | 0.476951 | 0.000 | 0.00000 | |

| | 50% | 75% | max |
|---|---|---|---|
| Unnamed: 0 | 384.5000 | 576.25000 | 768.00 |
| Pregnancies | 3.0000 | 6.00000 | 17.00 |
| Glucose | 117.0000 | 140.25000 | 199.00 |
| BloodPressure | 72.0000 | 80.00000 | 122.00 |

```
SkinThickness                23.0000    32.00000    99.00
Insulin                      30.5000   127.25000   846.00
BMI                          32.0000    36.60000    67.10
DiabetesPedigreeFunction      0.3725     0.62625     2.42
Age                          29.0000    41.00000    81.00
Outcome                       0.0000     1.00000     1.00
```

**Visualise Correlations between variables and particularly with respect to Outcome. Glucose, BMI and Age are 3 most correlated with Diabetes Outcome**

```
In [4]: corr = df.corr()
        pima_heatmap =sns.heatmap(corr, annot = True)
        pima_heatmap
        plt.savefig("heatmap.png");
```




# 1 Investigating the DataFrame for missing values.

```
In [5]: df=df.dropna(axis=0, how='any')
```

```
In [6]: df.describe().transpose()
```

Out[6]:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| Unnamed: 0 | 768.0 | 384.500000 | 221.846794 | 1.000 | 192.75000 | 384.5000 | 576.25000 | 768.00 |
| Pregnancies | 768.0 | 3.845052 | 3.369578 | 0.000 | 1.00000 | 3.0000 | 6.00000 | 17.00 |
| Glucose | 768.0 | 120.894531 | 31.972618 | 0.000 | 99.00000 | 117.0000 | 140.25000 | 199.00 |
| BloodPressure | 768.0 | 69.105469 | 19.355807 | 0.000 | 62.00000 | 72.0000 | 80.00000 | 122.00 |
| SkinThickness | 768.0 | 20.536458 | 15.952218 | 0.000 | 0.00000 | 23.0000 | 32.00000 | 99.00 |
| Insulin | 768.0 | 79.799479 | 115.244002 | 0.000 | 0.00000 | 30.5000 | 127.25000 | 846.00 |
| BMI | 768.0 | 31.992578 | 7.884160 | 0.000 | 27.30000 | 32.0000 | 36.60000 | 67.10 |
| DiabetesPedigreeFunction | 768.0 | 0.471876 | 0.331329 | 0.078 | 0.24375 | 0.3725 | 0.62625 | 2.42 |
| Age | 768.0 | 33.240885 | 11.760232 | 21.000 | 24.00000 | 29.0000 | 41.00000 | 81.00 |
| Outcome | 768.0 | 0.348958 | 0.476951 | 0.000 | 0.00000 | 0.0000 | 1.00000 | 1.00 |

No rows contain any NA's as number of rows remains 768

### 1.1 Investigating the DataFrame for null values (Pregnancies and Outcome will have values=0). All the others (except Insulin) would not be expected to contain 0 values so will be assumed to be missing values.

```
In [7]: (df==0).sum(axis=0)
```

```
Out[7]: Unnamed: 0                  0
        Pregnancies               111
        Glucose                     5
        BloodPressure              35
        SkinThickness             227
        Insulin                   374
        BMI                        11
        DiabetesPedigreeFunction    0
        Age                         0
        Outcome                   500
        dtype: int64
```

#### 1.1.1 About half of all subjects don't have Insulin values so column/variable dropped. It's possible that they do have 0 insulin value but then they would all be positive for diabetes and so this is not a possible explanation. Similarly approximately 30% (227/768) of Skin Thickness values are assumed missing as 0 and looking at the correlation with outcome (0.075), better to remove variable instead of dropping 227 samples to accommodate SkinThickness and use the median value for instance.

```
In [8]: df1=df.drop(['Insulin', 'SkinThickness'], axis=1)
```

```
In [9]: df1.describe().transpose()
```

Out[9]:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| Unnamed: 0 | 768.0 | 384.500000 | 221.846794 | 1.000 | 192.75000 | 384.5000 | 576.25000 | 768.00 |
| Pregnancies | 768.0 | 3.845052 | 3.369578 | 0.000 | 1.00000 | 3.0000 | 6.00000 | 17.00 |
| Glucose | 768.0 | 120.894531 | 31.972618 | 0.000 | 99.00000 | 117.0000 | 140.25000 | 199.00 |
| BloodPressure | 768.0 | 69.105469 | 19.355807 | 0.000 | 62.00000 | 72.0000 | 80.00000 | 122.00 |
| BMI | 768.0 | 31.992578 | 7.884160 | 0.000 | 27.30000 | 32.0000 | 36.60000 | 67.10 |
| DiabetesPedigreeFunction | 768.0 | 0.471876 | 0.331329 | 0.078 | 0.24375 | 0.3725 | 0.62625 | 2.42 |
| Age | 768.0 | 33.240885 | 11.760232 | 21.000 | 24.00000 | 29.0000 | 41.00000 | 81.00 |
| Outcome | 768.0 | 0.348958 | 0.476951 | 0.000 | 0.00000 | 0.0000 | 1.00000 | 1.00 |

**Now exclude subjects that contain 0 values in other variables**

```
In [10]: df1 = df1.loc[~((df1['Glucose'] == 0) | (df1['BloodPressure'] == 0) | (df1['BMI'] == 0)
```

**Now that null values have been removed Checking that the dataframe doesn't contain any other unusual values e.g min Age =1 or Glucose = 10. Also comparing mean of Outcome before and after to ensure that little or no change in Outcome value (0.348 before; 0.344 after)**

```
In [11]: df1.describe().transpose()
```

Out[11]:

| | count | mean | std | min | 25% |
|---|---|---|---|---|---|
| Unnamed: 0 | 724.0 | 385.781768 | 222.504870 | 1.000 | 192.750 |
| Pregnancies | 724.0 | 3.866022 | 3.362803 | 0.000 | 1.000 |
| Glucose | 724.0 | 121.882597 | 30.750030 | 44.000 | 99.750 |
| BloodPressure | 724.0 | 72.400552 | 12.379870 | 24.000 | 64.000 |
| BMI | 724.0 | 32.467127 | 6.888941 | 18.200 | 27.500 |
| DiabetesPedigreeFunction | 724.0 | 0.474765 | 0.332315 | 0.078 | 0.245 |
| Age | 724.0 | 33.350829 | 11.765393 | 21.000 | 24.000 |

```
Outcome                    724.0    0.343923    0.475344    0.000    0.000

                                50%       75%     max
Unnamed: 0                386.500  578.2500  768.00
Pregnancies                 3.000    6.0000   17.00
Glucose                   117.000  142.0000  199.00
BloodPressure              72.000   80.0000  122.00
BMI                        32.400   36.6000   67.10
DiabetesPedigreeFunction    0.379    0.6275    2.42
Age                        29.000   41.0000   81.00
Outcome                     0.000    1.0000    1.00
```

In [12]:
```
df1=df1.dropna(axis=0, how='any')
clean_df1=df1.copy()
y=clean_df1['Outcome'].copy()
y
```

Out[12]:
```
0      1
1      0
2      1
3      0
4      1
5      0
6      1
8      1
10     0
11     1
12     0
13     1
14     1
16     1
17     1
18     0
19     1
20     0
21     0
22     1
23     1
24     1
25     1
26     1
27     0
28     0
29     0
30     0
31     1
32     0
      ..
```

```
738    0
739    1
740    1
741    0
742    0
743    1
744    0
745    0
746    1
747    0
748    1
749    1
750    1
751    0
752    0
753    1
754    1
755    1
756    0
757    1
758    0
759    1
760    0
761    1
762    0
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 724, dtype: int64
```

In [13]:
```
variables = ['Pregnancies', 'Glucose', 'BloodPressure',  'BMI', 'DiabetesPedigreeFuncti
X = clean_df1[variables].copy()
X
```

Out[13]:

| | Pregnancies | Glucose | BloodPressure | BMI | DiabetesPedigreeFunction | Age |
|---|---|---|---|---|---|---|
| 0 | 6 | 148 | 72 | 33.6 | 0.627 | 50 |
| 1 | 1 | 85 | 66 | 26.6 | 0.351 | 31 |
| 2 | 8 | 183 | 64 | 23.3 | 0.672 | 32 |
| 3 | 1 | 89 | 66 | 28.1 | 0.167 | 21 |
| 4 | 0 | 137 | 40 | 43.1 | 2.288 | 33 |
| 5 | 5 | 116 | 74 | 25.6 | 0.201 | 30 |
| 6 | 3 | 78 | 50 | 31.0 | 0.248 | 26 |
| 8 | 2 | 197 | 70 | 30.5 | 0.158 | 53 |
| 10 | 4 | 110 | 92 | 37.6 | 0.191 | 30 |
| 11 | 10 | 168 | 74 | 38.0 | 0.537 | 34 |
| 12 | 10 | 139 | 80 | 27.1 | 1.441 | 57 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 13 | 1 | 189 | 60 | 30.1 | 0.398 | 59 |
| 14 | 5 | 166 | 72 | 25.8 | 0.587 | 51 |
| 16 | 0 | 118 | 84 | 45.8 | 0.551 | 31 |
| 17 | 7 | 107 | 74 | 29.6 | 0.254 | 31 |
| 18 | 1 | 103 | 30 | 43.3 | 0.183 | 33 |
| 19 | 1 | 115 | 70 | 34.6 | 0.529 | 32 |
| 20 | 3 | 126 | 88 | 39.3 | 0.704 | 27 |
| 21 | 8 | 99 | 84 | 35.4 | 0.388 | 50 |
| 22 | 7 | 196 | 90 | 39.8 | 0.451 | 41 |
| 23 | 9 | 119 | 80 | 29.0 | 0.263 | 29 |
| 24 | 11 | 143 | 94 | 36.6 | 0.254 | 51 |
| 25 | 10 | 125 | 70 | 31.1 | 0.205 | 41 |
| 26 | 7 | 147 | 76 | 39.4 | 0.257 | 43 |
| 27 | 1 | 97 | 66 | 23.2 | 0.487 | 22 |
| 28 | 13 | 145 | 82 | 22.2 | 0.245 | 57 |
| 29 | 5 | 117 | 92 | 34.1 | 0.337 | 38 |
| 30 | 5 | 109 | 75 | 36.0 | 0.546 | 60 |
| 31 | 3 | 158 | 76 | 31.6 | 0.851 | 28 |
| 32 | 3 | 88 | 58 | 24.8 | 0.267 | 22 |
| .. | ... | ... | ... | ... | ... | ... |
| 738 | 2 | 99 | 60 | 36.6 | 0.453 | 21 |
| 739 | 1 | 102 | 74 | 39.5 | 0.293 | 42 |
| 740 | 11 | 120 | 80 | 42.3 | 0.785 | 48 |
| 741 | 3 | 102 | 44 | 30.8 | 0.400 | 26 |
| 742 | 1 | 109 | 58 | 28.5 | 0.219 | 22 |
| 743 | 9 | 140 | 94 | 32.7 | 0.734 | 45 |
| 744 | 13 | 153 | 88 | 40.6 | 1.174 | 39 |
| 745 | 12 | 100 | 84 | 30.0 | 0.488 | 46 |
| 746 | 1 | 147 | 94 | 49.3 | 0.358 | 27 |
| 747 | 1 | 81 | 74 | 46.3 | 1.096 | 32 |
| 748 | 3 | 187 | 70 | 36.4 | 0.408 | 36 |
| 749 | 6 | 162 | 62 | 24.3 | 0.178 | 50 |
| 750 | 4 | 136 | 70 | 31.2 | 1.182 | 22 |
| 751 | 1 | 121 | 78 | 39.0 | 0.261 | 28 |
| 752 | 3 | 108 | 62 | 26.0 | 0.223 | 25 |
| 753 | 0 | 181 | 88 | 43.3 | 0.222 | 26 |
| 754 | 8 | 154 | 78 | 32.4 | 0.443 | 45 |
| 755 | 1 | 128 | 88 | 36.5 | 1.057 | 37 |
| 756 | 7 | 137 | 90 | 32.0 | 0.391 | 39 |
| 757 | 0 | 123 | 72 | 36.3 | 0.258 | 52 |
| 758 | 1 | 106 | 76 | 37.5 | 0.197 | 26 |
| 759 | 6 | 190 | 92 | 35.5 | 0.278 | 66 |
| 760 | 2 | 88 | 58 | 28.4 | 0.766 | 22 |
| 761 | 9 | 170 | 74 | 44.0 | 0.403 | 43 |
| 762 | 9 | 89 | 62 | 22.5 | 0.142 | 33 |
| 763 | 10 | 101 | 76 | 32.9 | 0.171 | 63 |
| 764 | 2 | 122 | 70 | 36.8 | 0.340 | 27 |
| 765 | 5 | 121 | 72 | 26.2 | 0.245 | 30 |

```
766            1      126             60  30.1                     0.349   47
767            1       93             70  30.4                     0.315   23

[724 rows x 6 columns]
```

```
In [14]: SEED = 42
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.25, random_state

         lr = LogisticRegression(random_state=SEED)
         knn = KNN()
         dt = DecisionTreeClassifier(random_state=SEED)
         classifiers = [('Logistic Regression', lr),('K Nearest Neighbours', knn),('Classificati
```

```
In [15]: for clf_name, clf in classifiers:
             clf.fit(X_train, y_train)
             y_pred = clf.predict(X_test)
             print('{:s} : {:.3f}'.format(clf_name, accuracy_score(y_test, y_pred)))
```

```
Logistic Regression : 0.773
K Nearest Neighbours : 0.768
Classification Tree : 0.762
```

```
In [16]: vc = VotingClassifier(estimators=classifiers)
         vc.fit(X_train, y_train)
         y_pred = vc.predict(X_test)
         print('Voting Classifier: {:.3f}'.format(accuracy_score(y_test, y_pred)))
```

```
Voting Classifier: 0.801
```

```
C:\Users\User\Anaconda3\lib\site-packages\sklearn\preprocessing\label.py:151: DeprecationWarning
  if diff:
```

```
In [17]: y_train.describe().transpose()
```

```
Out[17]: count    543.000000
         mean       0.357274
         std        0.479638
         min        0.000000
         25%        0.000000
         50%        0.000000
         75%        1.000000
         max        1.000000
         Name: Outcome, dtype: float64
```

```
In [18]: diabetes_classifier = DecisionTreeClassifier(max_leaf_nodes=10, random_state=42)
```

**Using Cross-Validation to try and minimise over-fitting where Training Mean Squared Error (MSE) is significantly lower than test MSE. CV =4 meaning that split into 4 groups and each time training takes place on 75% and test is 25%**

```
In [19]: MSE_CV = - cross_val_score(diabetes_classifier, X_train, y_train, cv= 4,
         scoring='neg_mean_squared_error',n_jobs = -1)

In [20]: diabetes_classifier.fit(X_train, y_train)

Out[20]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                     max_features=None, max_leaf_nodes=10,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, presort=False, random_state=42,
                     splitter='best')

In [21]: y_predict_train = diabetes_classifier.predict(X_train)

In [22]: y_predict_test = diabetes_classifier.predict(X_test)
```

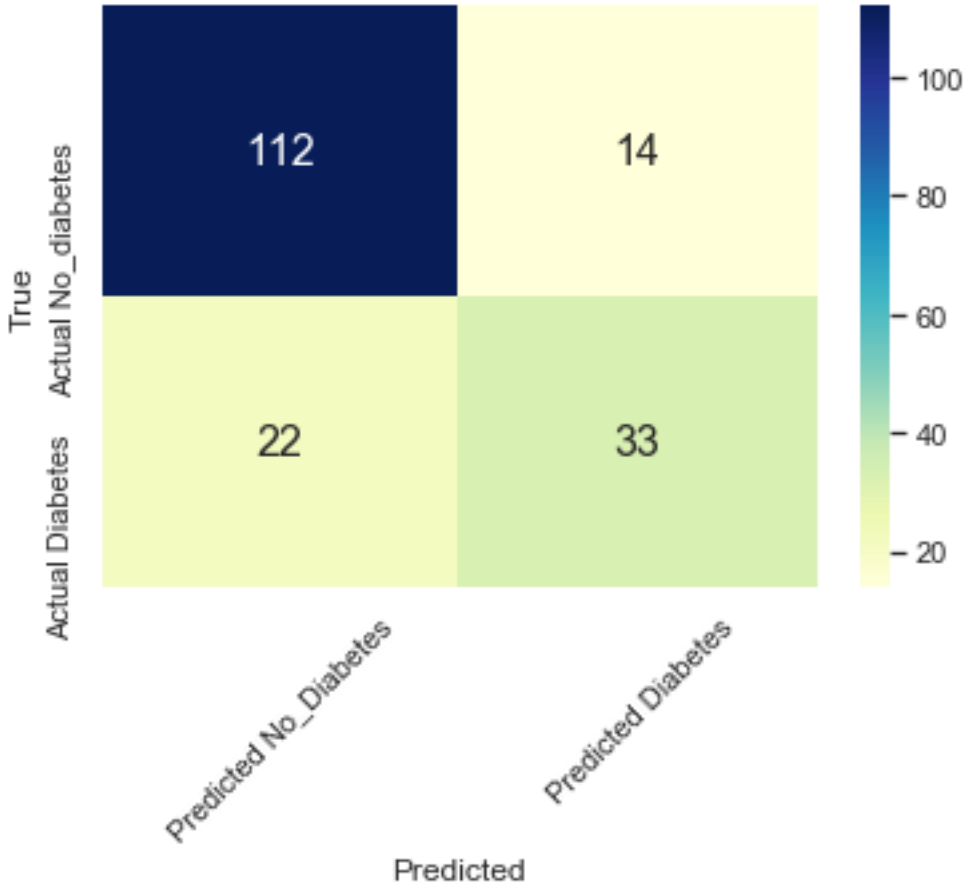
### 1.1.2 Determining the mean square error and ensuring that Train and Test MSE are similar.If test MSE is much larger than train MSE then this suggests that there is overfitting.

```
In [23]: print('CV : MSE {:.2f}'.format(MSE_CV.mean()))

CV : MSE 0.27


In [25]: print('Train MSE {:.2f}'.format(MSE(y_train, y_predict_train)))

Train MSE 0.18


In [24]: print('Test MSE {:.2f}'.format(MSE(y_test, y_predict_test)))

Test MSE 0.20


In [25]: predictions = diabetes_classifier.predict(X_test)

In [26]: accuracy_score(y_true = y_test, y_pred = predictions)

Out[26]: 0.8011049723756906
```

Confusion Matrix is True Neg False Pos False Neg True Pos

```
In [27]: cm = confusion_matrix(y_true = y_test, y_pred = predictions)
         cm

Out[27]: array([[112,  14],
                [ 22,  33]], dtype=int64)
```

In [28]:
```
array=[[112,  14],
       [ 22,  33]]
df_cm = pd.DataFrame(array,columns=["Predicted No_Diabetes", " Predicted Diabetes"], in
sns.set(font_scale=1)
sns.heatmap(df_cm, annot=True,cmap="YlGnBu", fmt=".0f",annot_kws={"size": 16})
plt.savefig("CM2.png")
plt.xlabel('Predicted')
plt.xticks(rotation=45)
plt.ylabel('True')
plt.ioff()
plt.show();
```




In [29]:
```
cr=classification_report(y_true = y_test, y_pred = predictions)
print(cr)
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.84 | 0.89 | 0.86 | 126 |
| 1 | 0.70 | 0.60 | 0.65 | 55 |

```
avg / total       0.80      0.80      0.80       181
```

**1.1.3 Precision is proportion of those identified as negative that were actually negative (112/(112+22)) = 0.84, Likewise for positives (33/(33+14)) = 0.70 Recall is proportion of those that were negative that were correctly identified as negative (112/(112+14)) = 0.89. For positive, (33/(33+22)) = 0.60**