Support Vector Machines - Breast Cancer Project

Nawal Obaid

9/7/2022

- Predicting if the cancer diagnosis is benign or malignant based on several observations/features
- 30 features are used, examples:

```
  - radius (mean of distances from center to points on the perimeter)
  - texture (standard deviation of gray-scale values)
  - perimeter
  - area
  - smoothness (local variation in radius lengths)
  - compactness (perimeter^2 / area - 1.0)
  - concavity (severity of concave portions of the contour)
  - concave points (number of concave portions of the contour)
  - symmetry
  - fractal dimension ("coastline approximation" - 1)
```

- Datasets are linearly separable using all 30 input features

- Number of Instances: 569
- Class Distribution: 212 Malignant, 357 Benign
- Target class:

```
   - Malignant
   - Benign
```

Import Needed Libraries

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

Load Data - Will be using a built in scikit-learn dataset called breast_cancer

In [2]:
```
from sklearn.datasets import load_breast_cancer
```

In [3]:
```
# need to instantiate an instance of this dataset
cancer = load_breast_cancer()
```

In [4]:
```
# we can explore this dataset
cancer.keys()
```

Out[4]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filenam
e', 'data_module'])
```

In [5]:
```
cancer['target_names']
# here need to use SVM as a classifier for two binary classes (0,1)
# 0 - Malignant, 1 - Benign
```

Out[5]:
```
array(['malignant', 'benign'], dtype='<U9')
```

In [6]:
```python
# if you need to get a full description of this dataset
# print(cancer['DESCR'])
```

In [7]:
```python
# create a dataframe out of this dataset
df = pd.DataFrame( np.c_[cancer['data'], cancer['target']],  columns= np.append(cancer['
```

In [8]:
```python
df.head()
```

Out[8]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	t
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	

5 rows × 31 columns

In [9]:
```python
df.info()
```

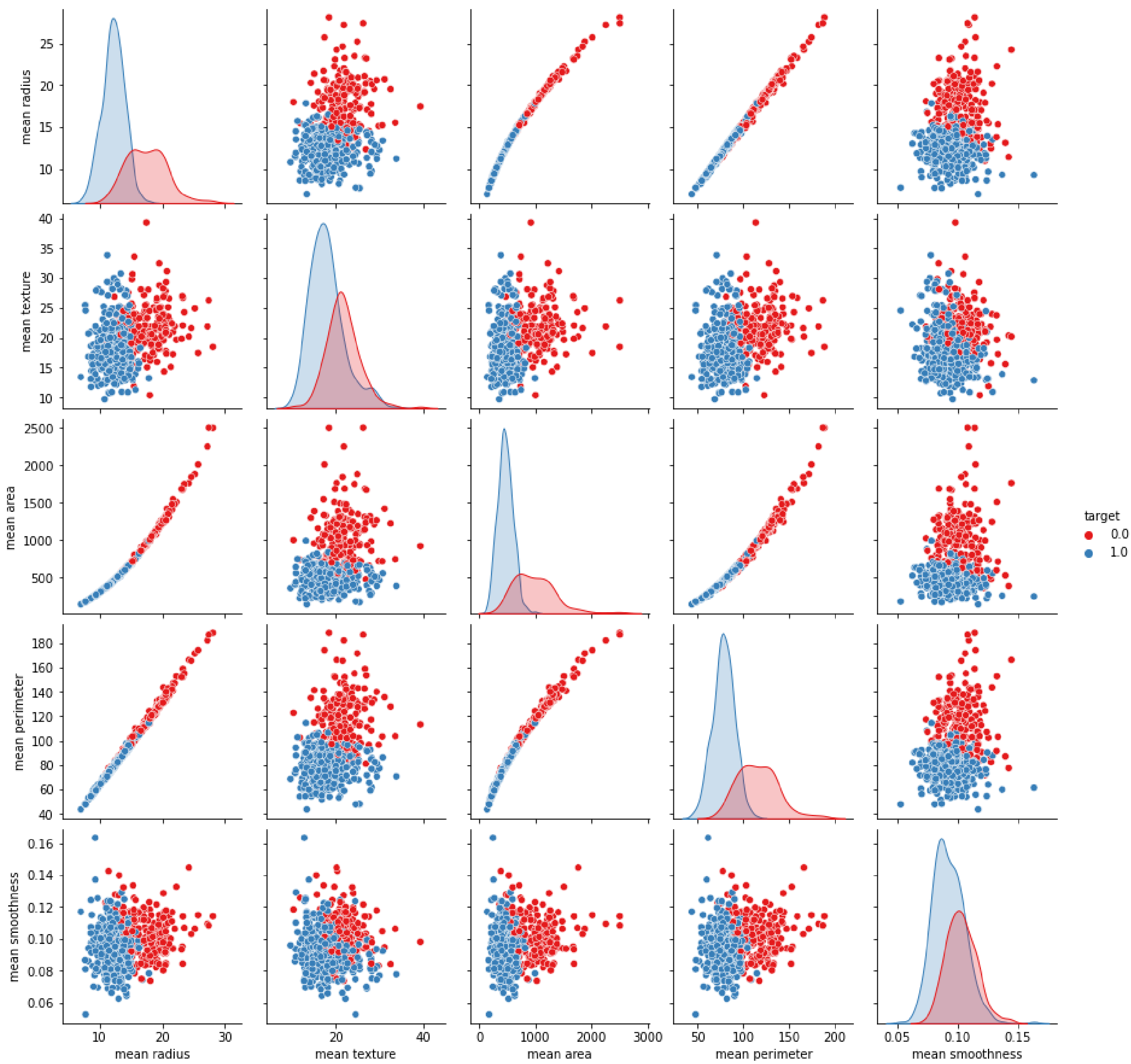
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
 30  target                   569 non-null    float64
```

```
dtypes: float64(31)
memory usage: 137.9 KB
```

EDA

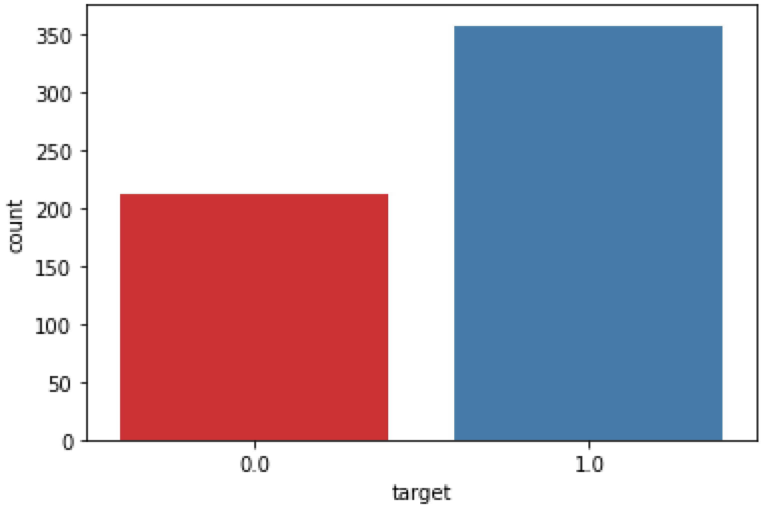
In [10]:
```
sns.pairplot(df, hue = 'target', vars = ['mean radius', 'mean texture', 'mean area', 'me
```

Out[10]:
```
<seaborn.axisgrid.PairGrid at 0x2622b4e2760>
```



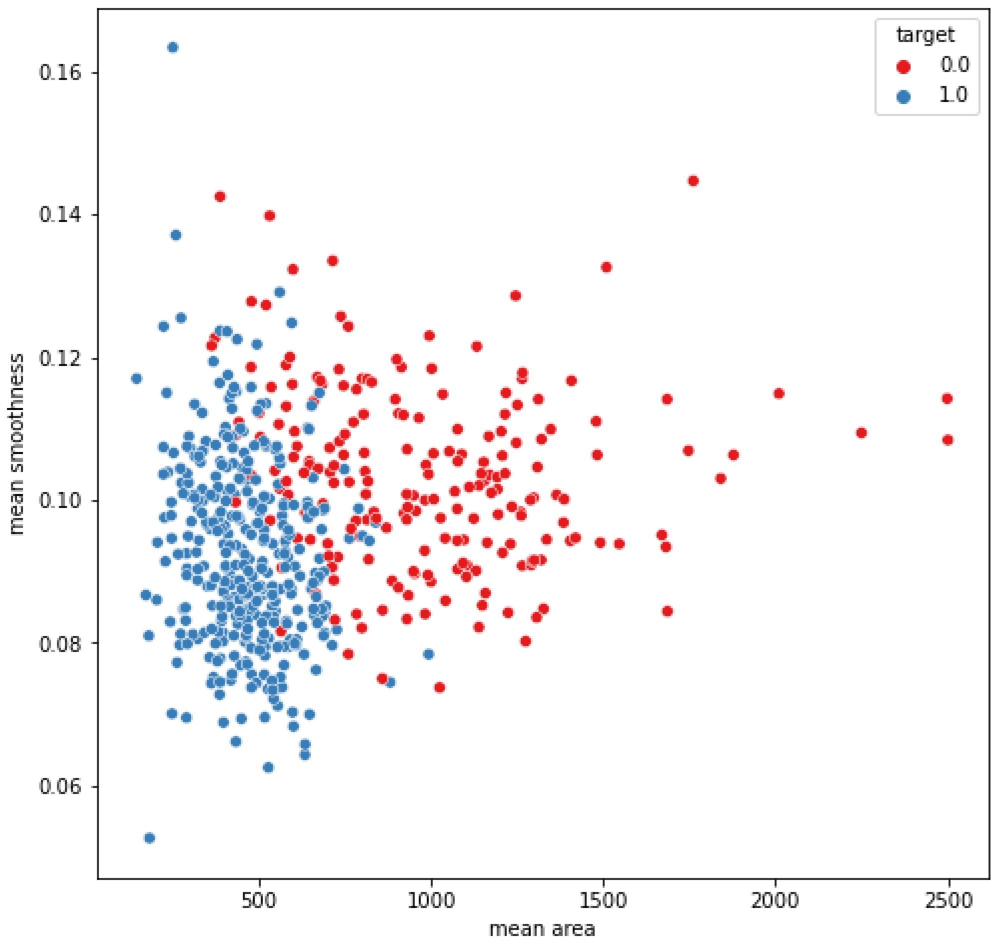
In [11]:
```
# let's count numver of benign and malignent cases
sns.countplot( x = df['target'], palette= 'Set1')

# Class Distribution: 212 - Malignant, 357 - Benign
```

Out[11]:
```
<AxesSubplot:xlabel='target', ylabel='count'>
```



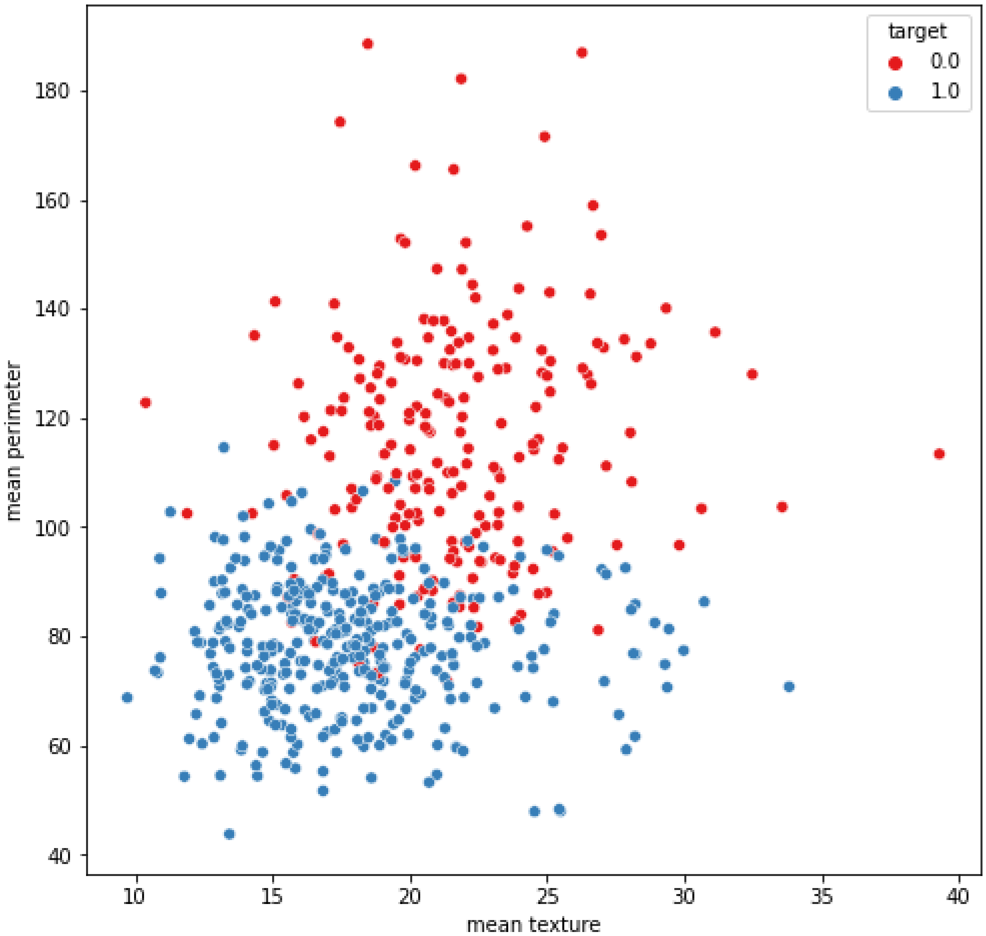
In [12]:
```
plt.figure(figsize=(8,8))
sns.scatterplot( data= df, x = df['mean area'], y = df['mean smoothness'], hue = df['tar
```

Out[12]:
```
<AxesSubplot:xlabel='mean area', ylabel='mean smoothness'>
```



In [13]:
```
plt.figure(figsize=(8,8))
sns.scatterplot( data= df, x = df['mean texture'], y = df['mean perimeter'], hue = df['t
```

Out[13]:
```
<AxesSubplot:xlabel='mean texture', ylabel='mean perimeter'>
```



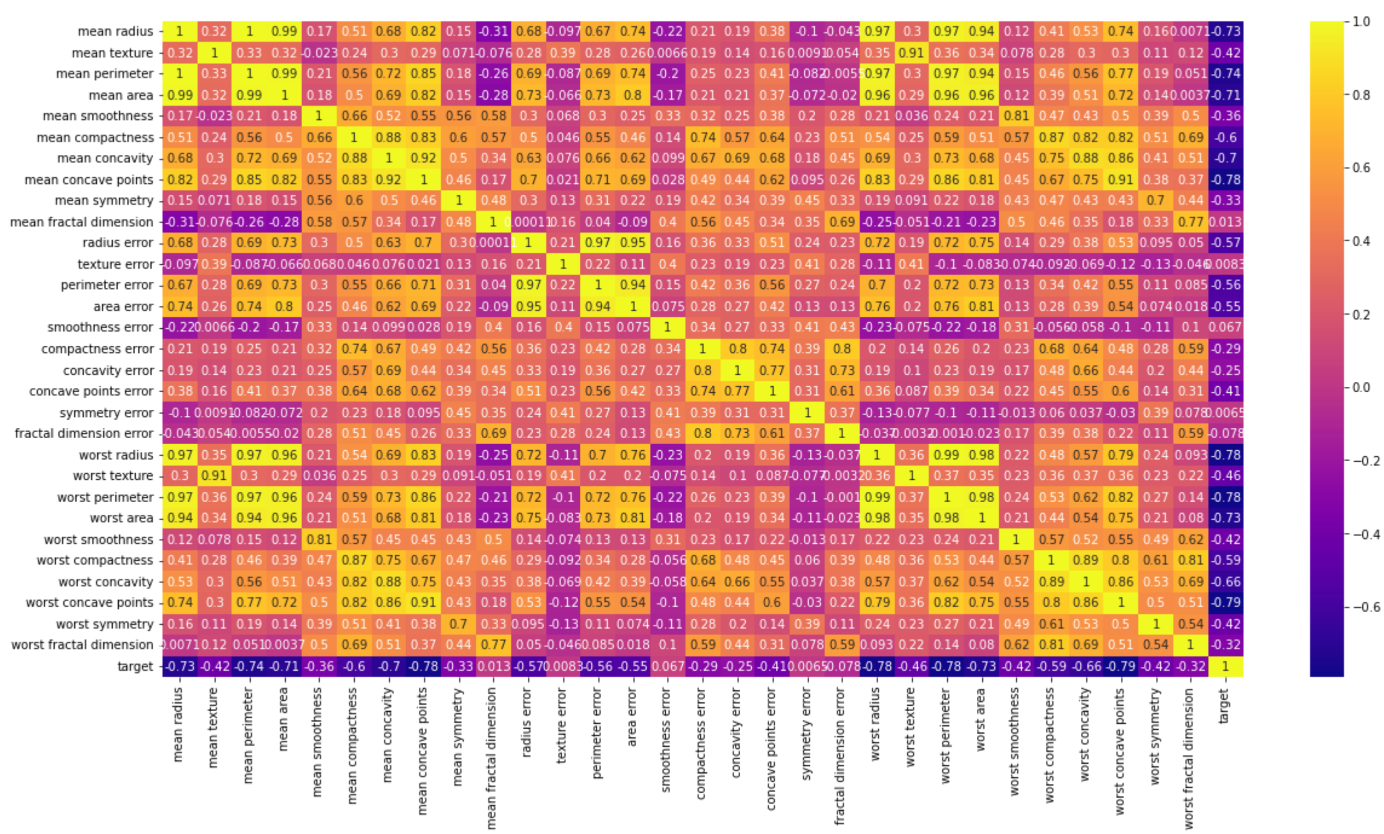
In [14]:

```
# we can also check the correlation among the variables
corr = df.corr()
```

In [15]:

```
plt.figure(figsize=(20,10))
sns.heatmap(corr, annot = True, cmap= 'plasma')
```

Out[15]: <AxesSubplot:>



Train The SVM Model

In [16]:
```
# recall that the df has all features--no target
X = df.drop(['target'], axis=1)
y = df['target']
```

In [17]:
```
from sklearn.model_selection import train_test_split
```

In [31]:
```
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2, random_state=1
```

In [32]:
```
# grab the Support Vector classifier
from sklearn.svm import SVC
classifier = SVC()
```

In [33]:
```
# now fit/train the model to the training set (X_train, y_train)
classifier.fit(X_train, y_train)
```

Out[33]:
```
▾ SVC
SVC()
```

In [34]:
```
# test/get pridections of the SVC model
predictions = classifier.predict(X_test)
```

In [35]:
```
# let's import the classifcation report and confusion matrix
from sklearn.metrics import classification_report, confusion_matrix
```

In [36]:
```
print('**************Classification Report*************************')
print(classification_report(y_test, predictions))
print('****************Confusion Matrix***************************')
print(confusion_matrix(y_test, predictions))
```

```
**************Classification Report**************************
              precision    recall  f1-score   support

         0.0       1.00      0.86      0.92        42
         1.0       0.92      1.00      0.96        72

    accuracy                           0.95       114
   macro avg       0.96      0.93      0.94       114
weighted avg       0.95      0.95      0.95       114

***************Confusion Matrix****************************
[[36  6]
 [ 0 72]]
```

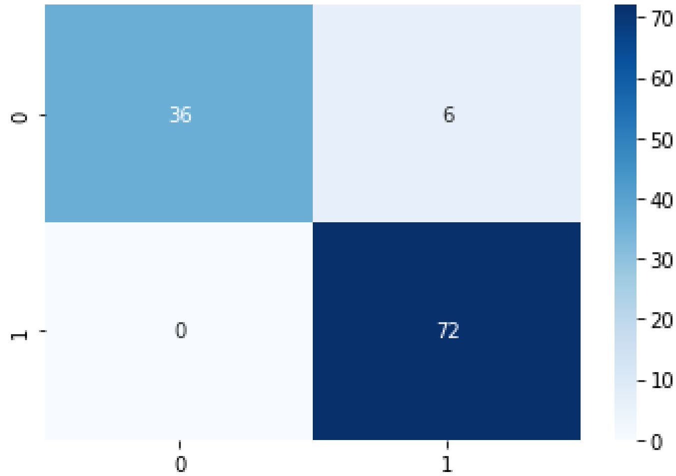
The confusion matrix shows:

False Positives(Type I Error)= 6

False Negatives(Type II Error)= 0

In [38]:
```
cm = confusion_matrix(y_test, predictions)
sns.heatmap(cm, annot = True, fmt = "d", cmap ='Blues')
```

Out[38]:
```
<AxesSubplot:>
```



36
6
0
72
0
1
0
1
70
60
50
40
30
20
10
0