```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_breast_cancer
%matplotlib inline

cancer = load_breast_cancer()
df_cancer = pd.DataFrame(np.c_[cancer['data'], cancer['target']], columns = np.app
df_cancer.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	r conca
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.

```
df_cancer.shape
```

```
(569, 31)
```

```
df_cancer.dtypes
```

```
mean radius                float64
mean texture               float64
mean perimeter             float64
mean area                  float64
mean smoothness            float64
mean compactness           float64
mean concavity             float64
mean concave points        float64
mean symmetry              float64
mean fractal dimension     float64
radius error               float64
texture error              float64
perimeter error            float64
area error                 float64
smoothness error           float64
compactness error          float64
concavity error            float64
concave points error       float64
symmetry error             float64
fractal dimension error    float64
worst radius               float64
worst texture              float64
worst perimeter            float64
worst area                 float64
```

```
worst smoothness           float64
worst compactness          float64
worst concavity            float64
worst concave points       float64
worst symmetry             float64
worst fractal dimension    float64
target                     float64
dtype: object
```

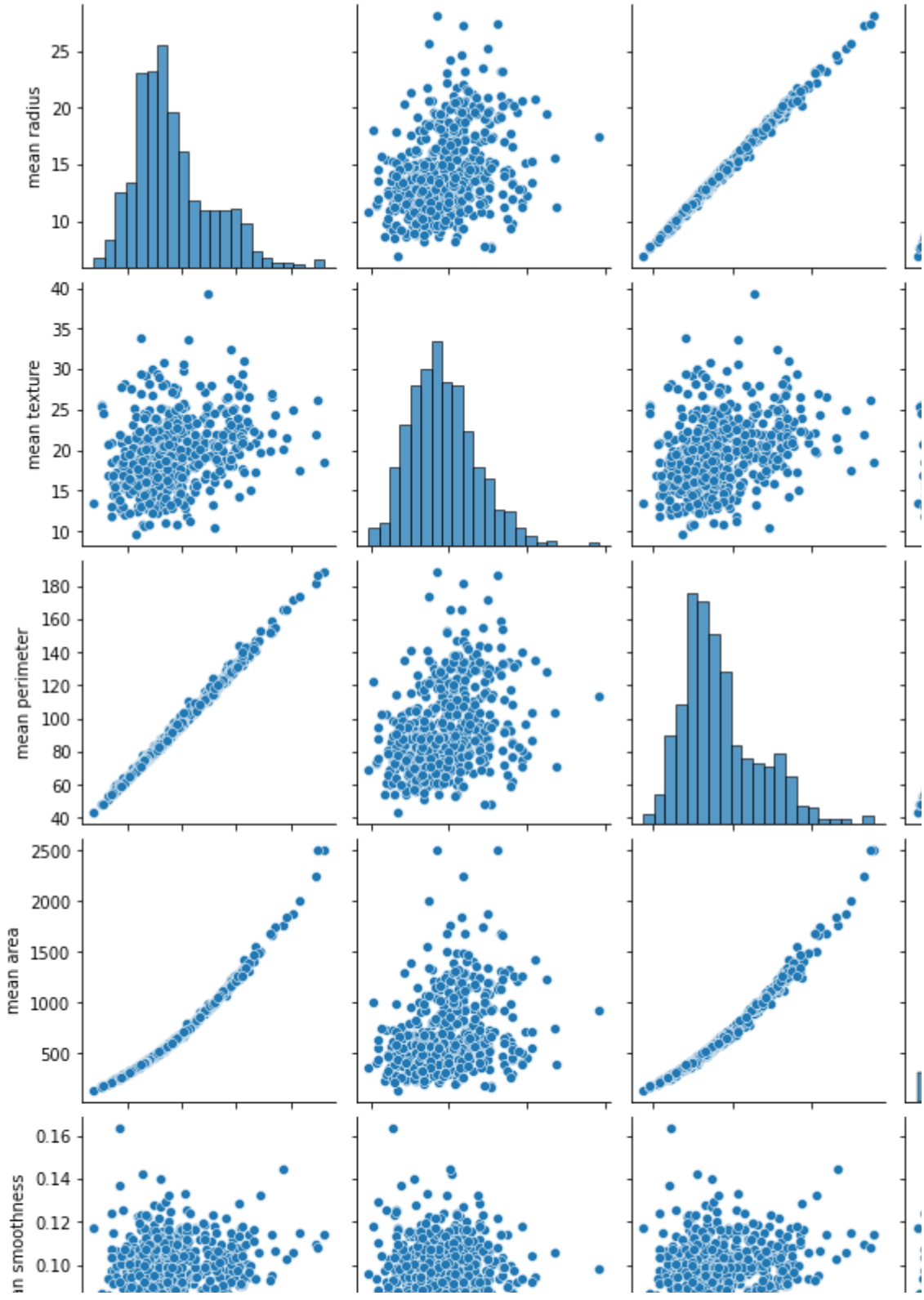
```
df_cancer.isna().sum()
```

```
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
symmetry error             0
fractal dimension error    0
worst radius               0
worst texture              0
worst perimeter            0
worst area                 0
worst smoothness           0
worst compactness          0
worst concavity            0
worst concave points       0
worst symmetry             0
worst fractal dimension    0
target                     0
dtype: int64
```

Visualize

```
# Let's plot out just the first 5 variables (features)
sns.pairplot(df_cancer, vars = ['mean radius', 'mean texture', 'mean perimeter', 'm
       'mean smoothness'] )
```

```
<seaborn.axisgrid.PairGrid at 0x7feba9adc350>
```

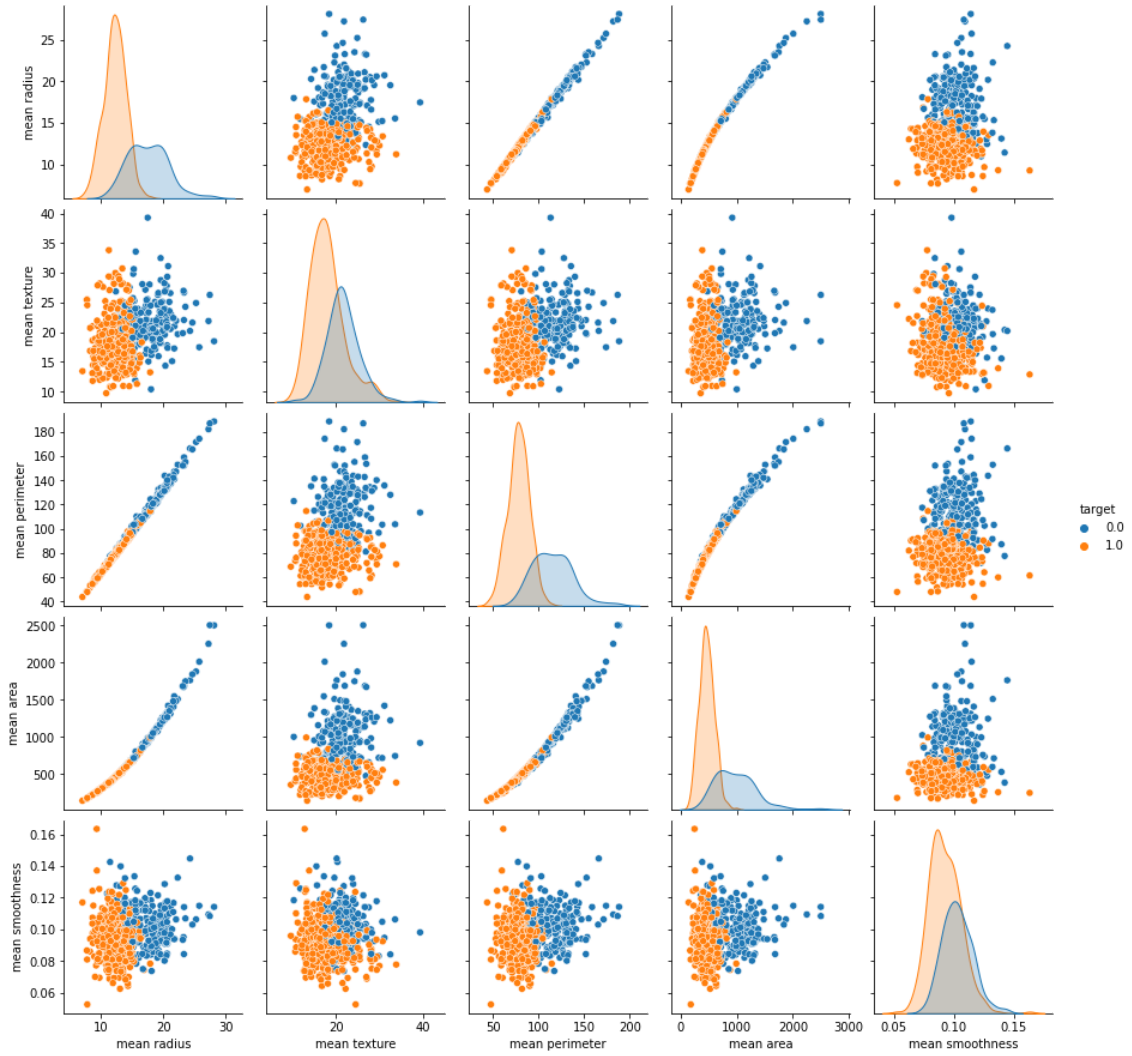


The above plots shows the relationship between our features. But the only problem with them is that they do not show us which of the "dots" is Malignant and which is Benign.

This issue will be addressed below by using "target" variable as the "hue" for the plots.

```
# Let's plot out just the first 5 variables (features)
sns.pairplot(df_cancer, hue = 'target', vars = ['mean radius', 'mean texture', 'mea
```

```
<seaborn.axisgrid.PairGrid at 0x7feb9f766c50>
```



Note:

1.0 (Orange) = Benign (No Cancer)

0.0 (Blue) = Malignant (Cancer)

Benign & Malignant count

```
df_cancer['target'].value_counts()
```

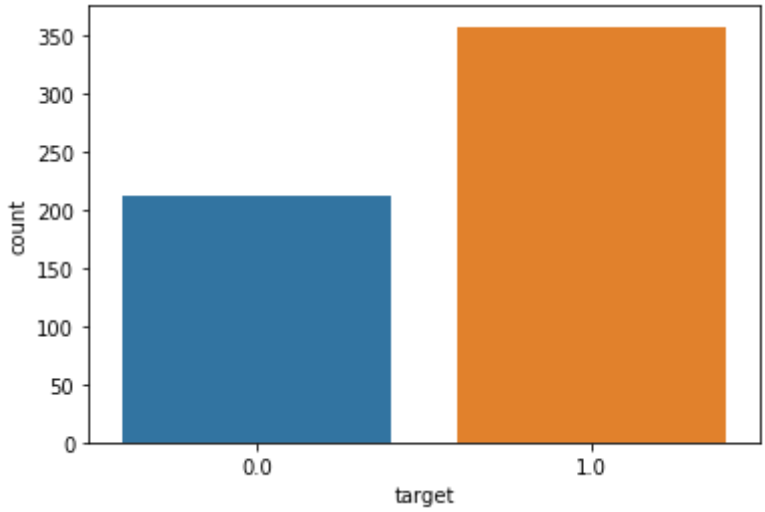
```
1.0    357
0.0    212
Name: target, dtype: int64
```

As we can see, we have 212 - Malignant, and 357 - Benign

Let's visulaize our counts

```
sns.countplot(df_cancer['target'], label = "Count")
```

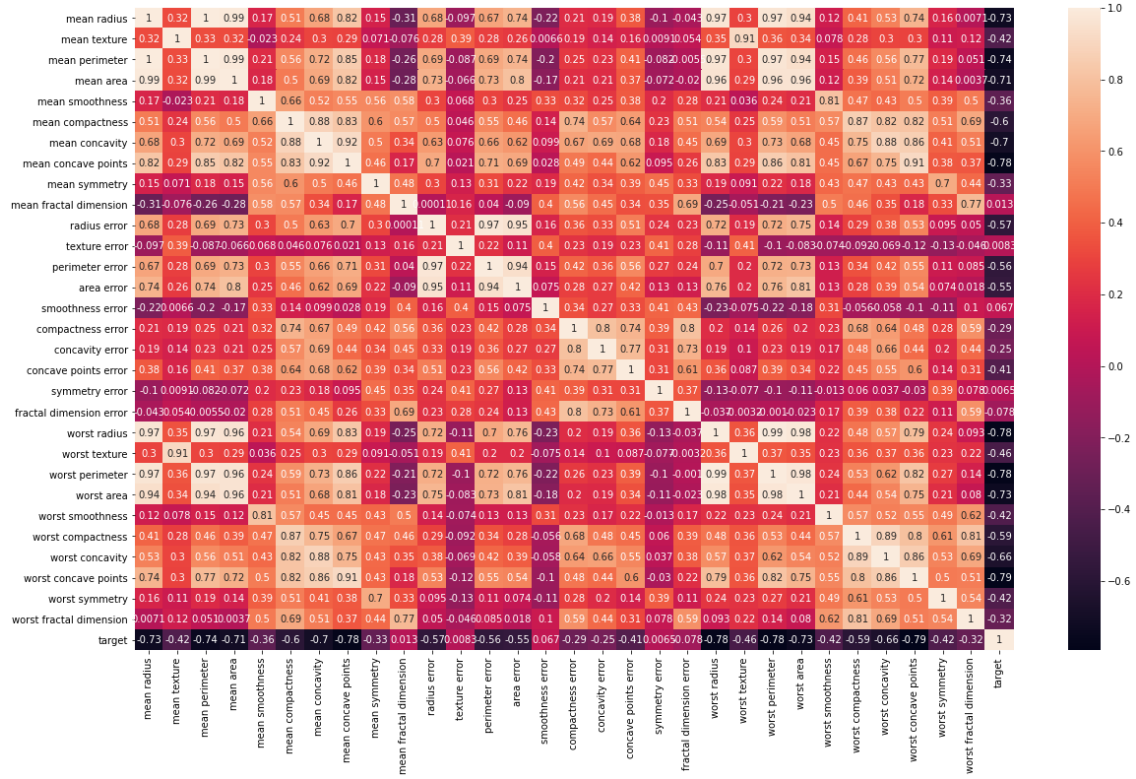
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: F
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7feb9b5c4910>
```



Let's check the correlation between our features

```
plt.figure(figsize=(20,12))
sns.heatmap(df_cancer.corr(), annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3355af5390>
```



There is a strong correlation between the mean radius and mean perimeter, mean area and mean primeter

Z-score Normalisation

```
X = df_cancer.drop(['target'], axis = 1) # We drop our "target" feature and use al
X.head()
```

```
          mean     mean       mean   mean        mean          mean        r
```

```
y = df_cancer['target']
y.head()
```

```
0    0.0
1    0.0
2    0.0
3    0.0
4    0.0
Name: target, dtype: float64
```

```
from sklearn.preprocessing import StandardScaler

X = np.array(X)
scaler = StandardScaler()
scaler.fit(X)
X_scaled = scaler.transform(X)


#Normalised dataframe
#-3 to +3
df_final = pd.DataFrame(X_scaled, columns = cancer['feature_names'])


df_final
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness
0	1.097064	-2.073335	1.269934	0.984375	1.568466	3.283515
1	1.829821	-0.353632	1.685955	1.908708	-0.826962	-0.487072
2	1.579888	0.456187	1.566503	1.558884	0.942210	1.052926
3	-0.768909	0.253732	-0.592687	-0.764464	3.283553	3.402909
4	1.750297	-1.151816	1.776573	1.826229	0.280372	0.539340
...	...	...	...	...	...	...
564	2.110995	0.721473	2.060786	2.343856	1.041842	0.219060
565	1.704854	2.085134	1.615931	1.723842	0.102458	-0.017833
566	0.702284	2.045574	0.672676	0.577953	-0.840484	-0.038680
567	1.838341	2.336457	1.982524	1.735218	1.525767	3.272144
568	-1.808401	1.221792	-1.814389	-1.347789	-3.112085	-1.150752

```
df_final.drop('mean perimeter',inplace=True,axis=1)
```

```
df_final
```

	mean radius	mean texture	mean area	mean smoothness	mean compactness	mean concavity
0	1.097064	-2.073335	0.984375	1.568466	3.283515	2.652874
1	1.829821	-0.353632	1.908708	-0.826962	-0.487072	-0.023846
2	1.579888	0.456187	1.558884	0.942210	1.052926	1.363478
3	-0.768909	0.253732	-0.764464	3.283553	3.402909	1.915897
4	1.750297	-1.151816	1.826229	0.280372	0.539340	1.371011
...	...	...	...	...	...	...
564	2.110995	0.721473	2.343856	1.041842	0.219060	1.947285
565	1.704854	2.085134	1.723842	0.102458	-0.017833	0.693043
566	0.702284	2.045574	0.577953	-0.840484	-0.038680	0.046588
567	1.838341	2.336457	1.735218	1.525767	3.272144	3.296944
568	-1.808401	1.221792	-1.347789	-3.112085	-1.150752	-1.114873

Suport Vector Maching (SVM)

What is a Support Vector Machine (SVM)?

A Support Vector Machine (SVM) is a binary linear classification whose decision boundary is explicitly constructed to minimize generalization error. It is a very powerful and versatile Machine Learning model, capable of performing linear or nonlinear classification, regression and even outlier detection.

SVM is well suited for classification of complex but small or medium sized datasets.

Model Training

Create the training and testing data

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(df_final, y, test_size = 0.2,
```

Import Support Vector Machine (SVM) Model

```
from sklearn.svm import SVC

svc_model = SVC()
```

Train

```
svc_model.fit(X_train, y_train)

    SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
        decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
        max_iter=-1, probability=False, random_state=None, shrinking=True,
        tol=0.001, verbose=False)
```

Prediction

```
y_predict = svc_model.predict(X_test)
```

	Predicted Negative	**Predicted Positive**
Actual Negative	True Negative (TN)	False Positive (FP)
Actual Positive	False Negative (FN)	True Positive (TP)

```
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, f1_s
confusion_matrix(y_test,y_predict)

    array([[47,  1],
           [ 0, 66]])

accuracy_score(y_test,y_predict)
```

```
0.9912280701754386
```

+ Code + Text

```
precision_score(y_test,y_predict)

    0.9850746268656716

recall_score(y_test,y_predict)

    1.0
```

```
f1_score(y_test,y_predict)
```

```
0.9924812030075187
```