Classification of Cancer Cells Using Machine Learning Model (SVM)

ARUNIMA VARMA

SEPTEMBER 22, 2020

Breast Cancer

- **Breast cancer** is the most common cancer amongst women in the world. It accounts for 25% of all cancer cases, and affected over 2.1 Million people in 2015 alone
- Early diagnosis significantly increases the chances of survival. The key challenges upon detection is how to classify tumours into malignant or benign.
- A tumour is considered malignant if the cells can grow into surrounding tissues or spread to distant areas of the body.
- A benign tumour does not invade nearby tissue nor spread to other parts of the body the way cancerous tumours can.

Machine Learning for Cancer Diagnosis

- Machine Learning is a sub-field of Artificial Intelligence that gives systems the ability to learn themselves without being explicitly programmed to do so. Machine Learning can be used in solving many real-world problems. ML techniques can dramatically improve the level of diagnosis in breast cancer. Research shows that experienced physicians can detect cancer by 79% accuracy, while a 91% accuracy can be achieved using ML techniques.

Data Acquisition and Cleaning

- Scikit-learn comes with a few small standard datasets that do not require downloading any file from any external website. The dataset that I will be using for our machine learning problem is the Breast cancer Wisconsin (diagnostic) dataset. The dataset includes several data about the breast cancer tumours along with the classification labels, viz., malignant or benign.

```
df_cancer.head()
```

Out[40]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	conc
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.1622	0.6656	0
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.1238	0.1866	0.
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.1444	0.4245	0.
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87	567.7	0.2098	0.8663	0.
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0	0.1374	0.2050	0.

5 rows × 31 columns

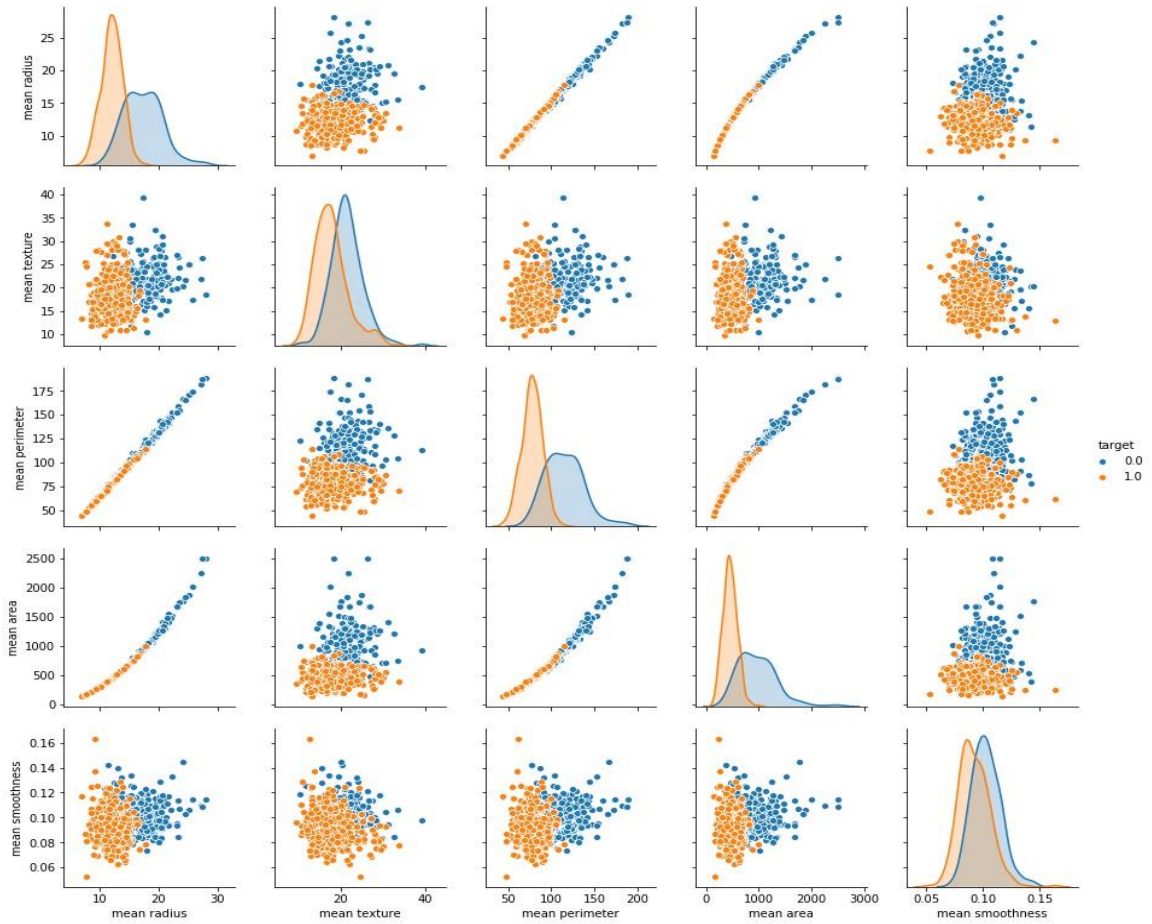
Feature Selection

- We see that each dataset of a tumour is labelled as either ‘malignant’ or ‘benign’. From here, each label is linked to binary values of 0 and 1, where 0 represents malignant tumours and 1 represents benign tumours.

```
# print the cancer labels (0:malignant, 1:benign)
print(cancer.target)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1
 1 0 1 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1
 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1 1
 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1
 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 1 0 1 1 0 1 0 1 0 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1]
```

Visualize the relationship between features



How many Benign and Malignant do we have in our dataset?

This can be visualized as follows:

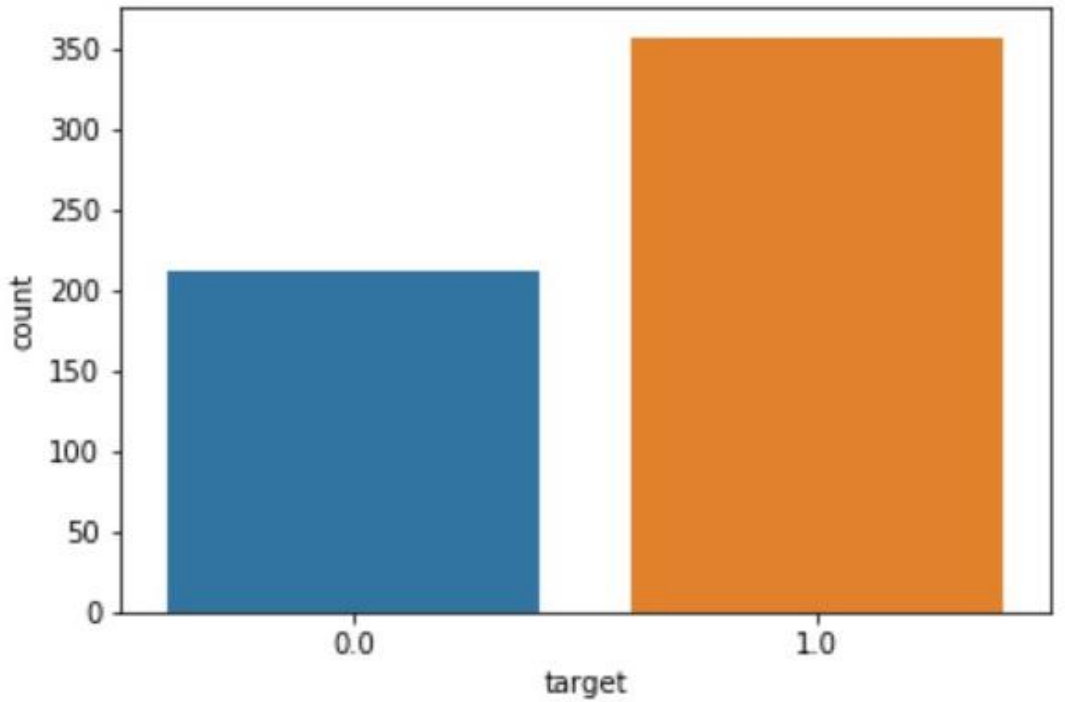
Note:

1.0 (Orange) = Benign (No Cancer)

0.0 (Blue) = Malignant (Cancer)

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

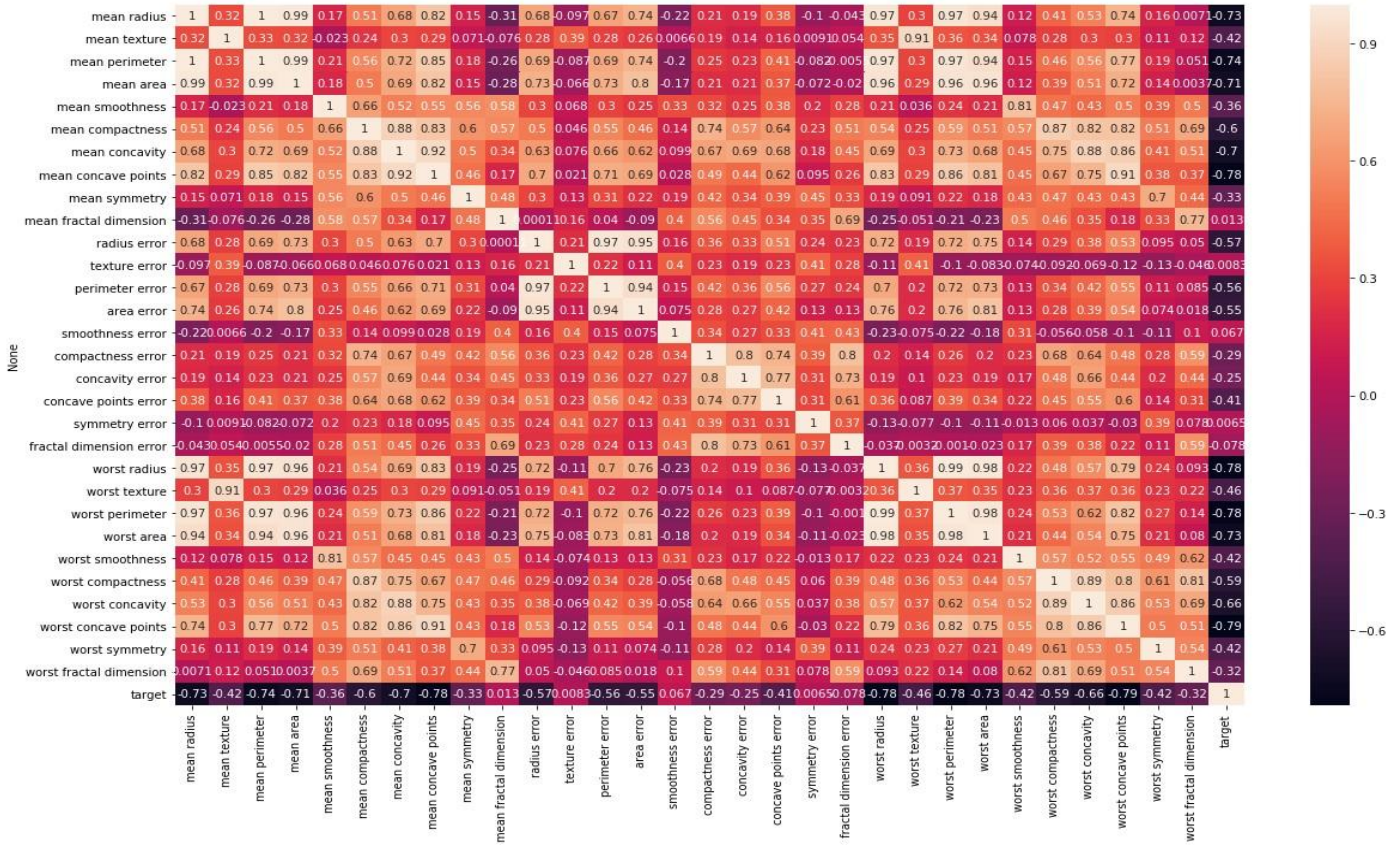
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6d439da396
```



Let's check the correlation between our features

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

```
15]: <matplotlib.axes._subplots.AxesSubplot at 0x7f6e3ecab320>
```



Introduction to Classification Modeling: Support Vector Machine (SVM)

- SVM offers very high accuracy compared to other classifiers such as logistic regression, and decision trees. It is known for its kernel trick to handle nonlinear input spaces. It is used in a variety of applications such as face detection, intrusion detection, classification of emails, news articles and web pages, classification of genes, and handwriting recognition.
- SVM is an exciting algorithm and the concepts are relatively simple. The classifier separates data points using a hyperplane with the largest amount of margin. That's why an SVM classifier is also known as a discriminative classifier. SVM finds an optimal hyperplane which helps in classifying new data points.

Generating Model

To build support vector machine model:
First, import the SVM module and create support vector classifier object by passing argument kernel as the linear kernel in SVC() function.
Then, fit the model on train set using fit() and perform prediction on the test set using predict().

Then, fit the model on train set using fit() and perform prediction on the test set using predict().

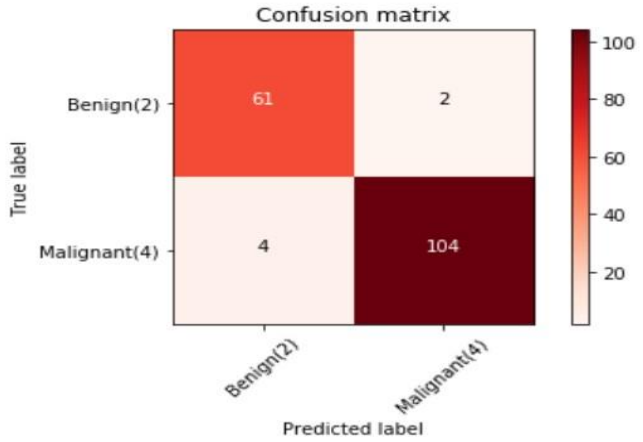
17]:
```
#Import svm model
from sklearn import svm

#Create a svm Classifier
clf = svm.SVC(kernel='linear') # Linear Kernel

#Train the model using the training sets
clf.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

Evaluating the Model

Confusion Matrix:

	precision	recall	f1-score	support
0	0.94	0.97	0.95	63
1	0.98	0.96	0.97	108
micro avg	0.96	0.96	0.96	171
macro avg	0.96	0.97	0.96	171
weighted avg	0.97	0.96	0.97	171

```
Confusion matrix, without normalization
[[ 61   2]
 [  4 104]]
```



F1_Score and Jaccard Index:

We can also use the **f1_score** from sklearn library:

```
om sklearn.metrics import f1_score
int("f1_score is :",f1_score(y_test, y_pred, average='weighted'))
```

```
f1_score is : 0.9650224422036399
```

We'll try **jaccard index** for accuracy:

```
om sklearn.metrics import jaccard_similarity_score
int("jaccard index is: ",jaccard_similarity_score(y_test, y_pred))
```

```
jaccard index is:  0.9649122807017544
```

The advantages and Disadvantages of SVM

Advantages:

- Effective in high dimensional spaces.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function.

Disadvantages:

- If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial.
- SVMs do not directly provide probability estimates.

Conclusion

- This project took us through the journey of explaining what “modelling” means in Data Science, An introduction to Support Vector Machine (SVM), advantages and disadvantages of SVM, Training an SVM model to make accurate breast cancer classifications, Evaluating the performance of an SVM model, and testing model accuracy using Confusion Matrix.