Gaussian NB Supervised Machine Learning

March 23, 2021

[424]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import scipy.stats as stats
import numpy as np
```

[425]:
```
data = pd.read_csv('iris.csv')
```

[426]:
```
data.head()
```

[426]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

[427]:
```
print('n_Features:',data.shape[1]-1)
```

```
n_Features: 5
```

[428]:
```
print('n_Samples:',data.shape[0])
```

```
n_Samples: 150
```

[429]:
```
#Feature matrix, the features that are not used for labeling(eg what we are␣
 ↪wanting to predict)
feature_matrix = data.iloc[:,0:-1]
feature_matrix = feature_matrix.drop(columns='Id')
fm = feature_matrix

#Target array, consists of the labels of our sample that we are going to try␣
 ↪and create a model to predict
target_array = data['Species']
ta = target_array
```

[430]:
```
ta.head()
```

[430]:
```
0    Iris-setosa
1    Iris-setosa
2    Iris-setosa
3    Iris-setosa
4    Iris-setosa
Name: Species, dtype: object
```

0.1 Cleaning

[431]:
```
data.isna().sum()
```

[431]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

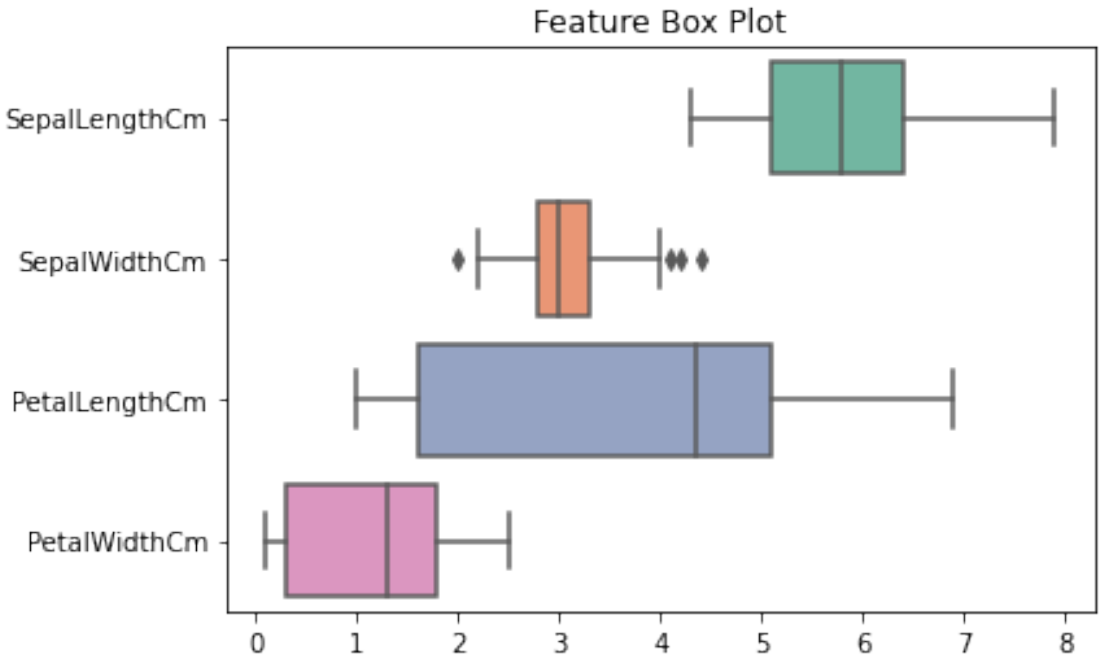
[432]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

0.1.1 Investigating Outliers

All features (exluding Id which is not a features) are level 4 **Ration level** data.

[433]:
```
ax = sns.boxplot(data=data.iloc[:,1:], orient="h", palette="Set2")
plt.title('Feature Box Plot')
plt.show()
```



[436]:
```
z = stats.zscore(fm)
abs_z_scores = np.abs(z)
fe =(abs_z_scores <3).all(axis=1)
fm = fm[fe]
ta = ta[fe]
```

Depending on our accuracy, we can try removing outliers later.

[437]:
```
fm.corr()
```

[437]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109370	0.875204	0.819851
SepalWidthCm	-0.109370	1.000000	-0.409417	-0.347337
PetalLengthCm	0.875204	-0.409417	1.000000	0.962598
PetalWidthCm	0.819851	-0.347337	0.962598	1.000000

ussally a high correlation could suggest redundant features. Which could suggest the need for redundant feature removal, however based on the small amount of features already present in our feature matrix this may not be benifitial. In this data sets case all the columns are highly related to each other, this may not necicarilly be a bad thing however.

0.2 Analysis of correlation between plant species

[438]:
```
species= data['Species'].unique()
species = pd.Series(species)
columns = data.iloc[:,1:-1].columns
cols = pd.Series(['sl','sw','pl','pw'])

sp_corr = {}

for c,s in enumerate(species):
    holder = data[data['Species']==s]
    holder = holder.reset_index()
    holder = holder.drop(columns='index')
    sp_corr[f'{s}'] = holder.iloc[:,1:-1]
```

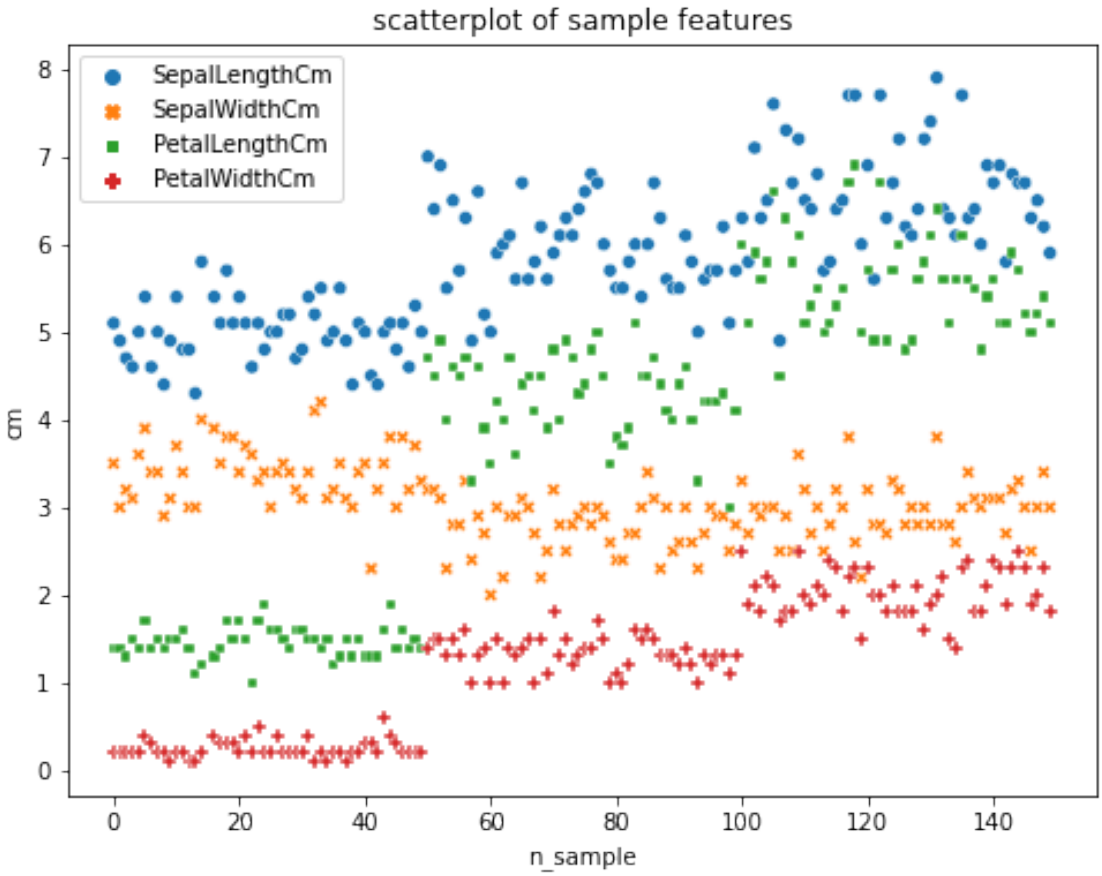
[439]:
```
print(sp_corr[species[0]].corrwith(sp_corr[species[1]]))
print(sp_corr[species[1]].corrwith(sp_corr[species[2]]))
print(sp_corr[species[2]].corrwith(sp_corr[species[0]]))
```

```
SepalLengthCm   -0.080850
SepalWidthCm    -0.102924
PetalLengthCm   -0.188226
PetalWidthCm    -0.189826
dtype: float64
SepalLengthCm   -0.141666
SepalWidthCm    -0.086514
PetalLengthCm   -0.087821
PetalWidthCm     0.009845
dtype: float64
SepalLengthCm    0.134172
SepalWidthCm    -0.004418
PetalLengthCm    0.100933
PetalWidthCm     0.119766
dtype: float64
```

0.3 Preprocessing

there are only 4 features so feature reduction is not nessicary, This ML model will be Supervised Learning. And due to the fact that our labels/target array prediction are decrete i will be using a **Classification** algorithim.

[440]:
```
fig = plt.figure(figsize=(8,6))
sns.scatterplot(data=fm)
plt.title('scatterplot of sample features')
plt.xlabel('n_sample')
plt.ylabel('cm')
plt.show()
```

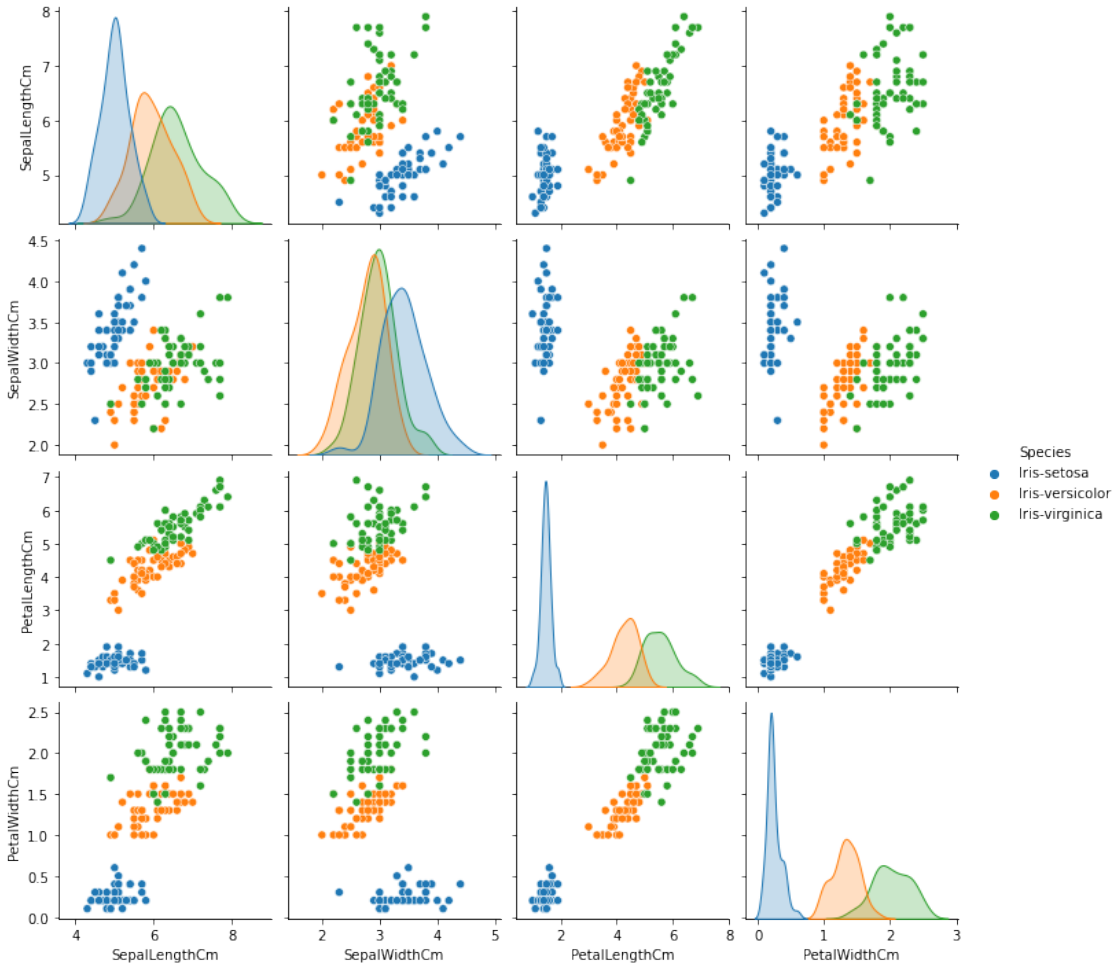


Although its not all that terrible here, the difference in scales between features can still be seen. for example the range of cm values for petalwidth is much smaller than that of sepal length. we will need to standardize all these feautes to prevent favoratizim in our model.

Luckly all the values are using the same metric CM so we dont need to do any conversions.

[415]:
```
sns.pairplot(data=data.iloc[:,1:],hue='Species')
```

[415]:
```
<seaborn.axisgrid.PairGrid at 0x216eb8aec10>
```



0.3.1 Scaling

[497]:
```
scaler = preprocessing.StandardScaler().fit(fm)
fm_scaled = scaler.transform(fm)
x_iris = pd.DataFrame(fm_scaled)
```

[498]:
```
xtrain, xtest, ytrain, ytest =train_test_split(x_iris,ta,test_size=0.
↪25,random_state=10)
```

Here we can already see the emergence of the groupings based on features and species

[499]:
```
model_ = GaussianNB()
model_.fit(xtrain,ytrain)
y_model=model_.predict(xtest)
```

[500]:
```
accuracy_score(ytest,y_model)
```

[500]: 0.9473684210526315

[ ]: