Imporintg packages

```
import seaborn as sns
import pandas as pd
import numpy as np
import altair as alt
import sklearn as sklearn
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
```

Loading and inspecting dataset

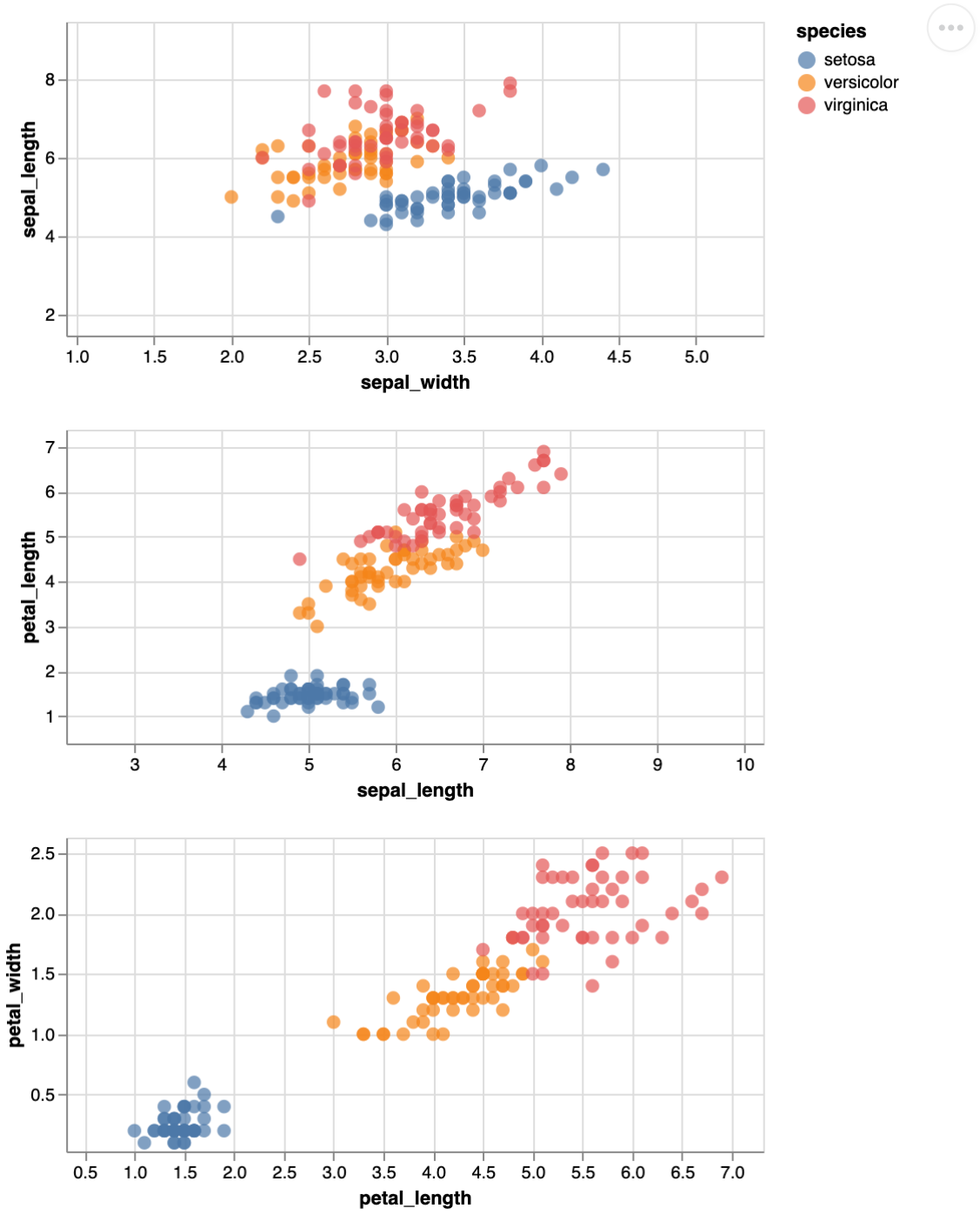
```
iris = sns.load_dataset('iris')
print(iris.head())
print(type(iris))
print(iris['species'].value_counts())
```

```
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
<class 'pandas.core.frame.DataFrame'>
versicolor    50
virginica     50
setosa        50
Name: species, dtype: int64
```

Visualizing data

```
a = alt.Chart(iris).mark_circle(size=60).encode(
    x='sepal_width',
    y='sepal_length',
    color='species'
).interactive().properties(width=400,height=180)
b = alt.Chart(iris).mark_circle(size=60).encode(
    x='sepal_length',
    y='petal_length',
    color='species'
).interactive().properties(width=400,height=180)
c = alt.Chart(iris).mark_circle(size=60).encode(
    x='petal_length',
```

```
    y='petal_width',
    color='species'
).interactive().properties(width=400,height=180)
alt.vconcat(a, b,c)
```



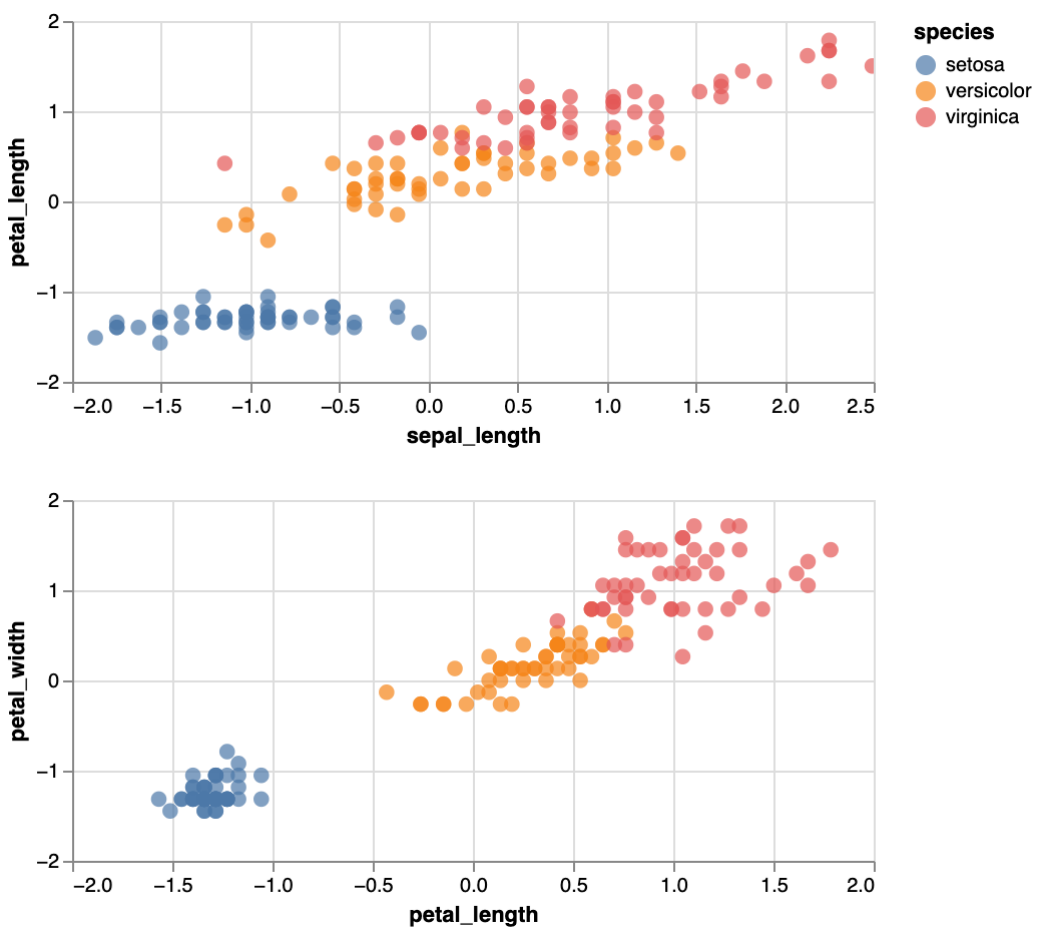
Normalizing data

```
iris2 = iris.iloc[:,0:4]
iris2.iloc[:,0:4] = StandardScaler().fit_transform(iris2)
iris2['species'] = iris.species
```

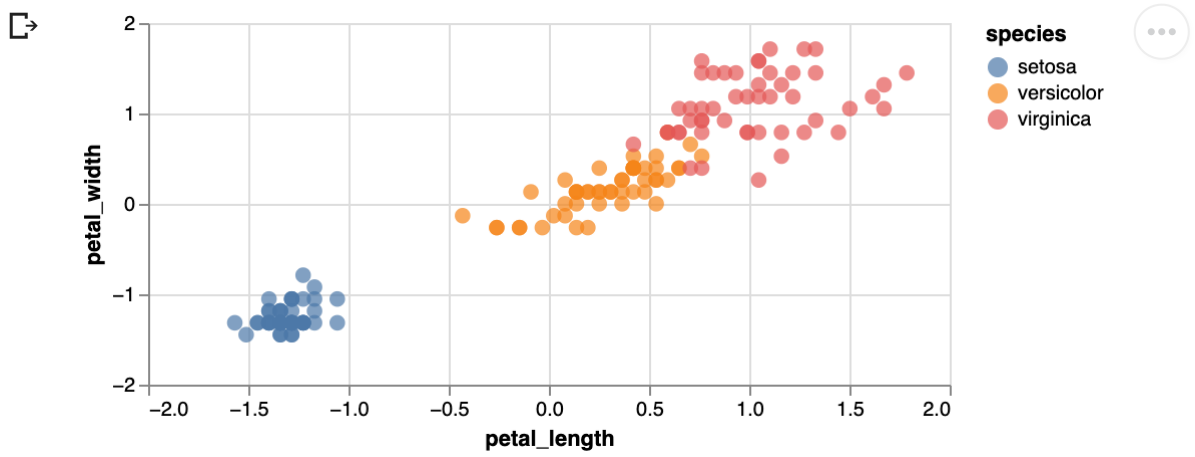
Plotting normalized data

```
a = alt.Chart(iris2).mark_circle(size=60).encode(
    x='sepal_width',
    y='sepal_length',
    color='species'
).interactive().properties(width=400,height=180)
b = alt.Chart(iris2).mark_circle(size=60).encode(
    x='sepal_length',
    y='petal_length',
    color='species'
).interactive().properties(width=400,height=180)

alt.vconcat(a,b)
```




```
c = alt.Chart(iris2).mark_circle(size=60).encode(
    x='petal_length',
    y='petal_width',
    color='species'
).interactive().properties(width=400,height=180)
c
```



Is feature scaling helpful for this dataset?

Scaling helped spread the datapoints further apart from each other, which might be helpful for the dataset.

Run classification with/without scaling.

Calculating performance too.

```
clf = GaussianNB().fit(iris.iloc[:,0:4], iris.species)
clf.score(iris.iloc[:,0:4], iris.species)

clf2 = GaussianNB().fit(iris2.iloc[:,0:4], iris2.species)
clf2.score(iris2.iloc[:,0:4], iris2.species)
```

```
0.96
```

What if we use only two features (intead of all 4)?

```
clf3 = GaussianNB().fit(iris.iloc[:,[2,3]], iris.species)
clf3.score(iris.iloc[:,[2,3]], iris.species)

clf4 = GaussianNB().fit(iris2.iloc[:,[2,3]], iris2.species)
clf4.score(iris2.iloc[:,[2,3]], iris2.species)
```

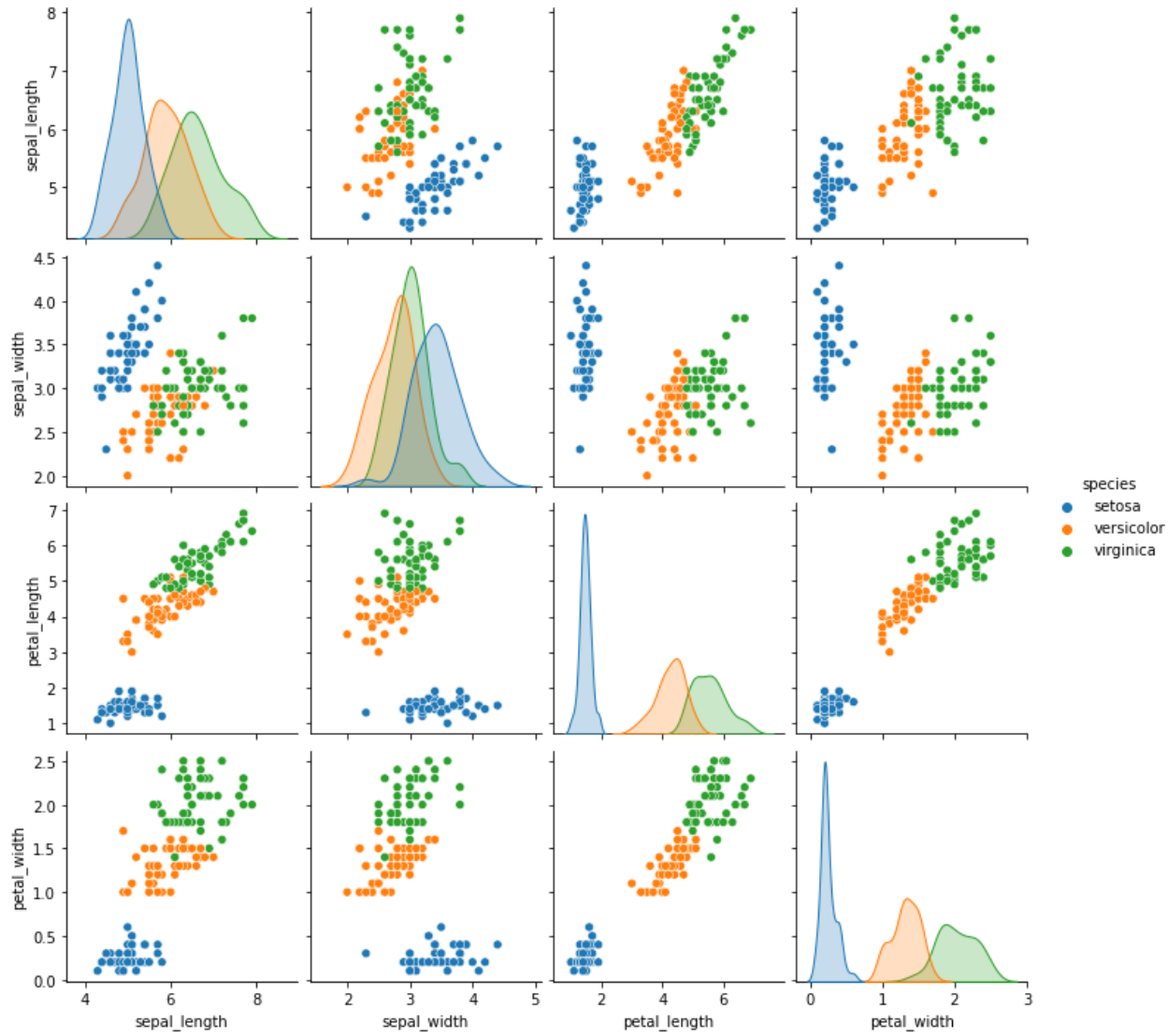
```
0.96
```

Performance of 2 features is similar to performance of 4 features.

Plotting classification for non-normalized data

```
iris3 = iris.copy()
iris3['species'] = clf.predict(iris.iloc[:,0:4])
sns.pairplot(iris3, hue="species")
```

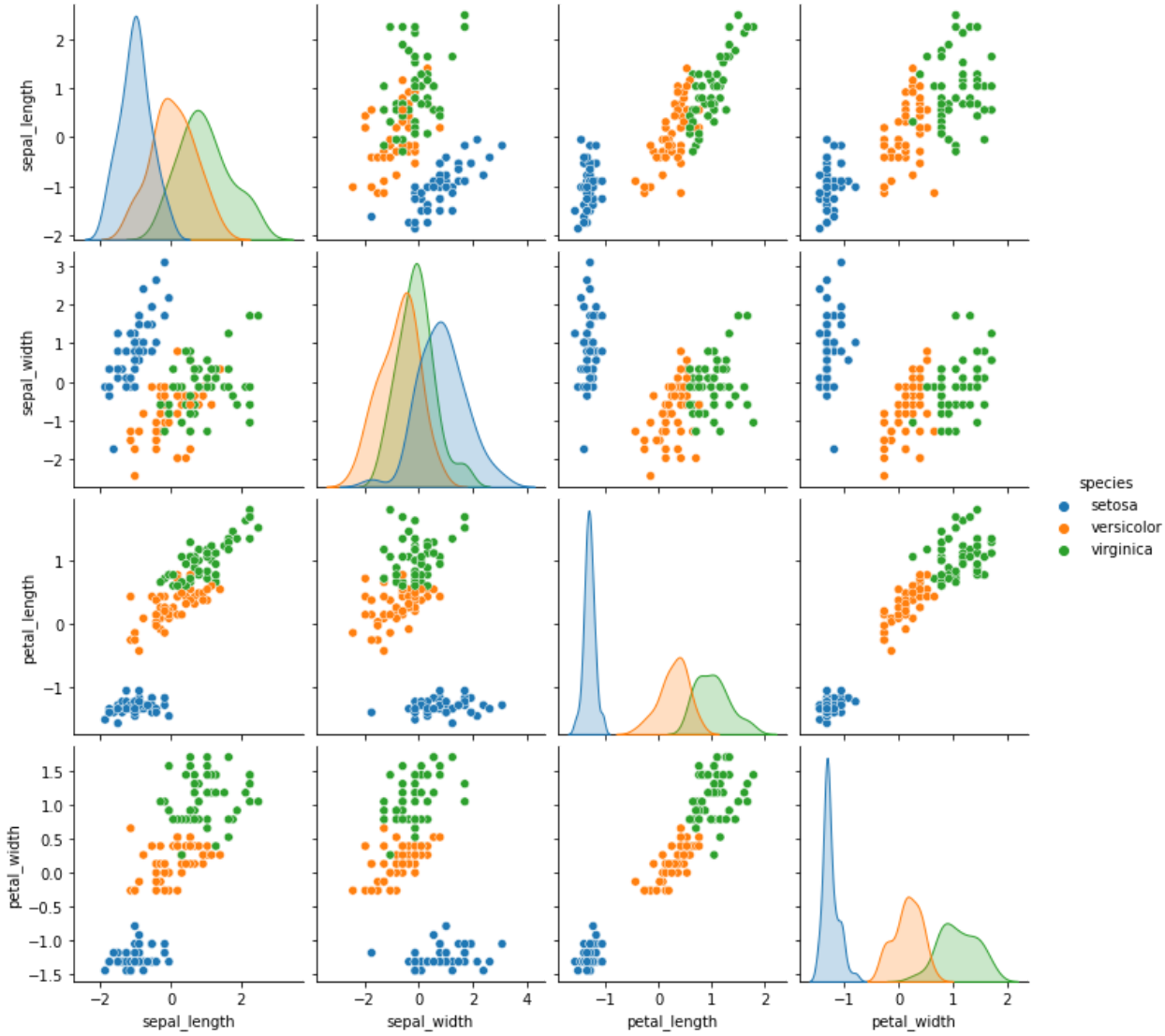
```
<seaborn.axisgrid.PairGrid at 0x7f2da4cf5b50>
```



Plotting classification for normalized data

```
iris4 = iris2.copy()
iris4['species'] = clf2.predict(iris2.iloc[:,0:4])
sns.pairplot(iris4, hue="species")
```

```
<seaborn.axisgrid.PairGrid at 0x7f2da4c804d0>
```



Pretty much the same.

✓ 0s completed at 10:51 AM