DAT405 Assignment 2 - Group 52

Hampus Jansson - (6 hrs)

Erik Johannesen - (6 hrs)

April 11, 2022

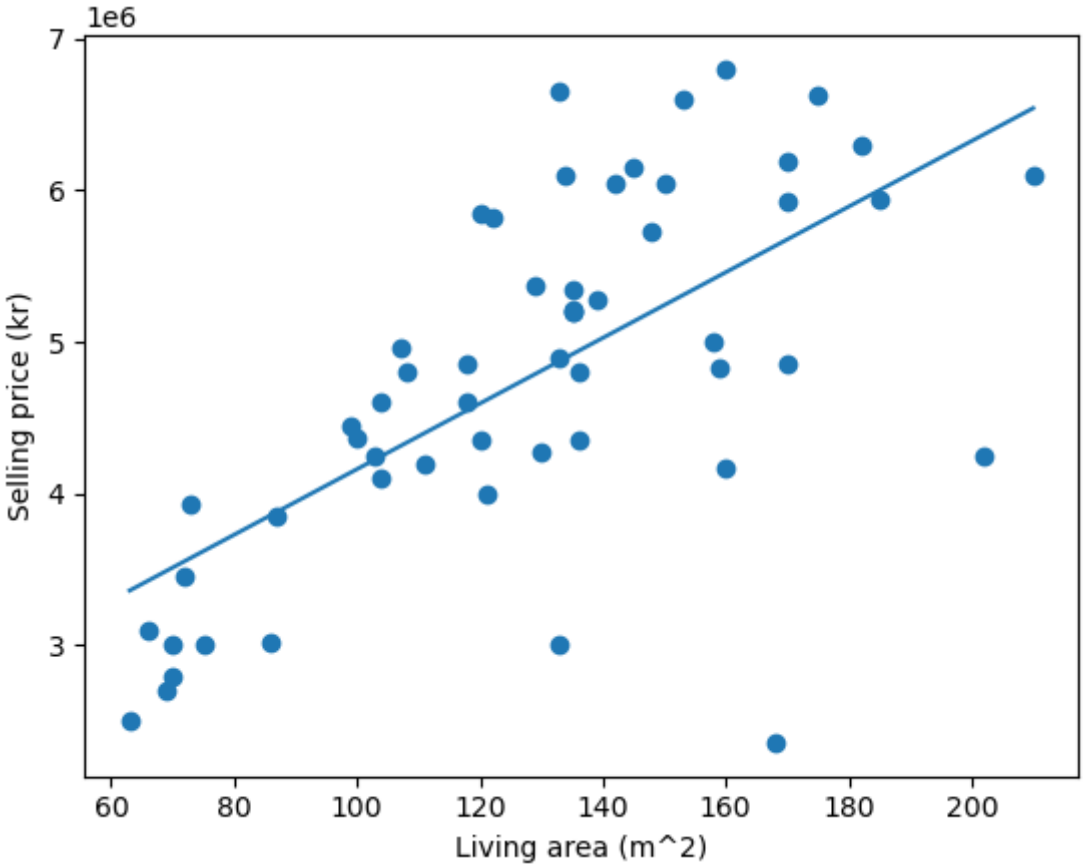
Problem 1

a)

Beneath is shown a linear regression model that relates the living area to the selling price for villas in Landvetter [2].

The data cleaning involved removing entries that did not have data for landsize which might have been because it was not a villa.

In [18]:
```
import pandas
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

#reading dataframe
df1 = pandas.read_csv("C:\\Users\\kalla\\DataJupyter\\data_assignment2.csv")

#filtering out entities without landsize
df1 = df1[(df1['ID'] != 41)]

#creating scatterplot
plt.scatter(df1['Living_area'], df1['Selling_price'])
plt.xlabel('Living area (m^2)')
plt.ylabel('Selling price (kr)')


l_price = df1['Living_area'].to_numpy()
#creating linear regression model and showing it together with the scatterplot
model = LinearRegression()
model.fit(l_price[:, np.newaxis], df1['Selling_price'].to_numpy())

xfit = np.array([63, 210])
yfit = model.predict(xfit[:, np.newaxis])

plt.plot(xfit, yfit)
plt.show()
```



b)

Below are the values of the intersection and slope presented.

In [19]:
```
#Printing the values of the intersection and slope.
print("The intersection value is: "+ str(model.intercept_))
print("The value of the slope is: "+ str(model.coef_[0]))
```

```
The intersection value is: 1999337.623725114
The value of the slope is: 21631.101576907153
```

c)

Presented below is the predicted selling prices for the living areas; 10, 100, 150, 200 and 1000 square meters using the model.

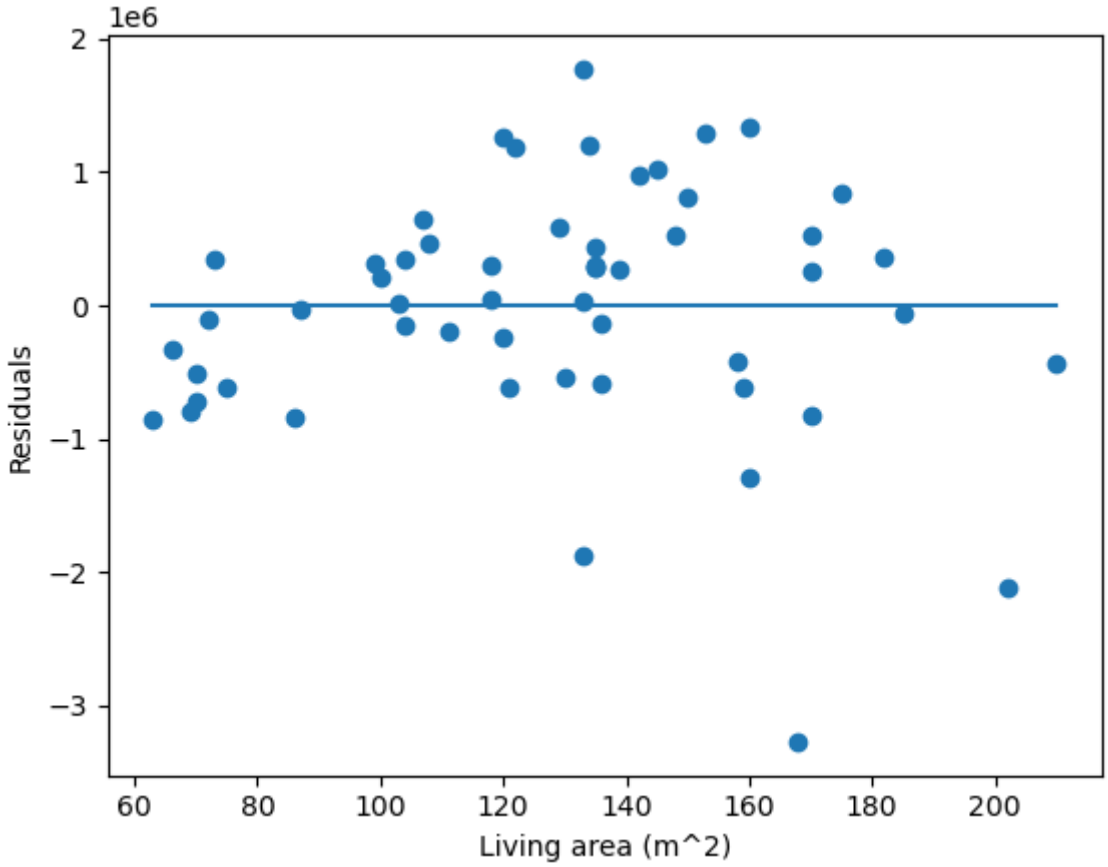
In [20]:
```
#inputting values for house sizes into the linear regrssion model to predict their
xVals = np.array([10, 100, 150, 200, 1000])
yVals = model.predict(xVals[:, np.newaxis])


for i, val in enumerate(xVals):
    print("The price of a villa with a living space of: " + str(val) + " is: " + s
```

```
The price of a villa with a living space of: 10 is: 2215648.6394941853
The price of a villa with a living space of: 100 is: 4162447.781415829
The price of a villa with a living space of: 150 is: 5244002.860261187
The price of a villa with a living space of: 200 is: 6325557.9391065445
The price of a villa with a living space of: 1000 is: 23630439.200632267
```

d)

A residual plot for the entries based on the linear regression model created.

In [25]:
```
#calculating residuals
yPredicted2 = model.predict(l_price[:, np.newaxis])
residuals = [df1['Selling_price'].to_numpy()[j]-yPredicted2[j] for j in range(len(
plt.scatter(df1['Living_area'], residuals)
livA = df1['Living_area']
sellP = df1['Selling_price']

plt.xlabel('Living area (m^2)')
plt.ylabel('Residuals')

#creating residuals plot
xvalues = (np.min(livA),np.max(livA))
yvalues = (0, 0)
plt.plot(xvalues, yvalues)

#show plot
plt.show()
```



e)

The residuals are unbalanced and the variance is high, which indicates that the model is not useful.

To improve the model it should take into account other measurements aswell, such as land size and biarea.

A similar approach could be more useful in areas other than Landvetter that does not have a lot of land area around the villas. The exact same line could not be used though, the values would have to be adjusted for the new area.

Problem 2

a) Visualization

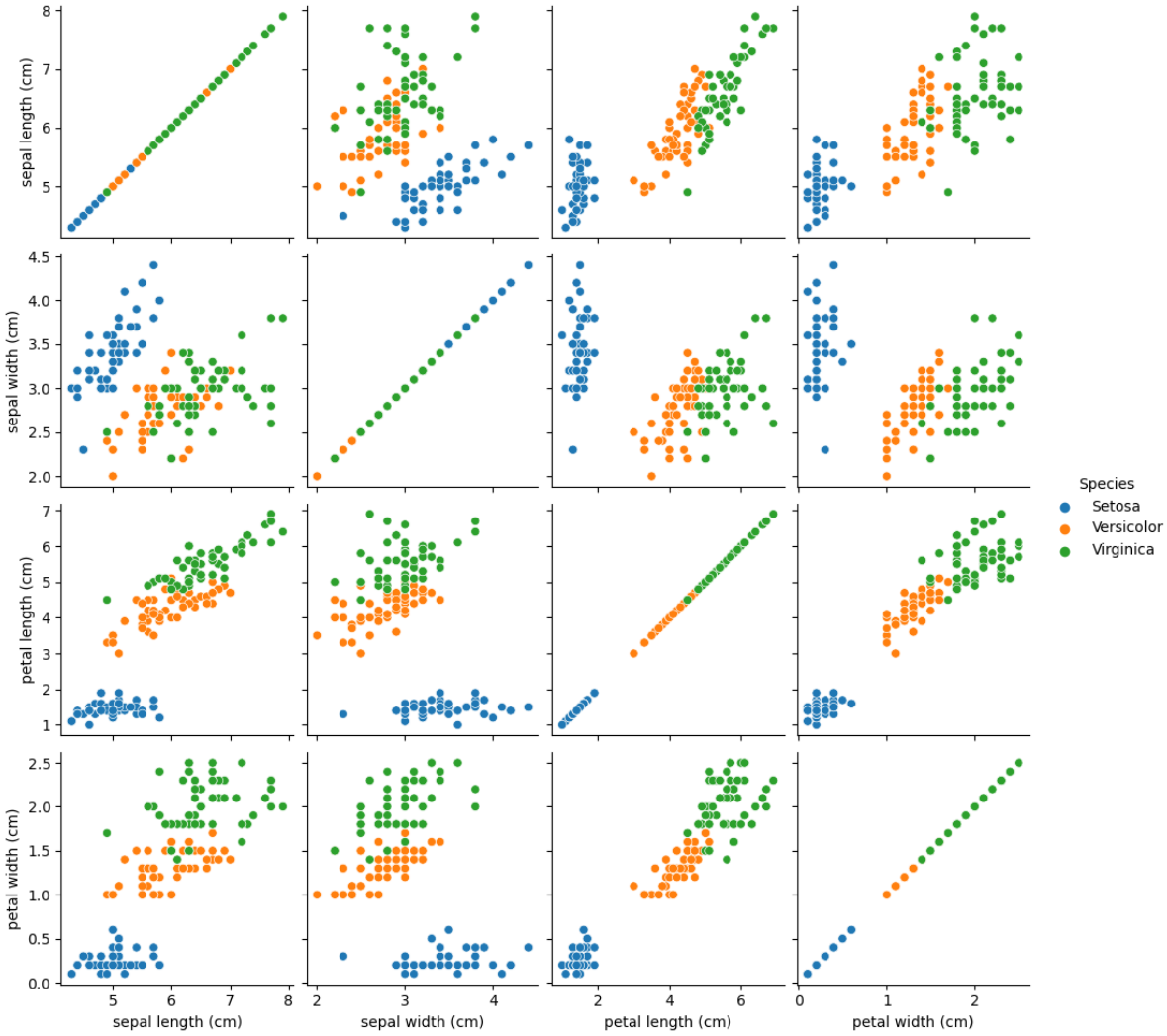
The data used is about different species of irises [2].

The data shows that the species clearly differs between one another.

It also shows that the Iris Setosa differs the most from the other species and that it differs more from Iris Virginica than the Iris Versicolor.

In [22]:
```
from sklearn.datasets import load_iris
from sklearn import svm, neighbors
from sklearn.model_selection import train_test_split
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from matplotlib.colors import ListedColormap
from sklearn.inspection import DecisionBoundaryDisplay
import seaborn as sns

#reading dataframe
data = load_iris(as_frame=True)['data']

#adding a new column with information about the species
species = np.repeat(['Setosa','Versicolor','Virginica'], 50)
data['Species'] = species

#creating scatterplots for all the different combinations of variables
sns.pairplot(data, hue = 'Species', diag_kind = None)
```

Out[22]:
```
<seaborn.axisgrid.PairGrid at 0x263493c66b0>
```

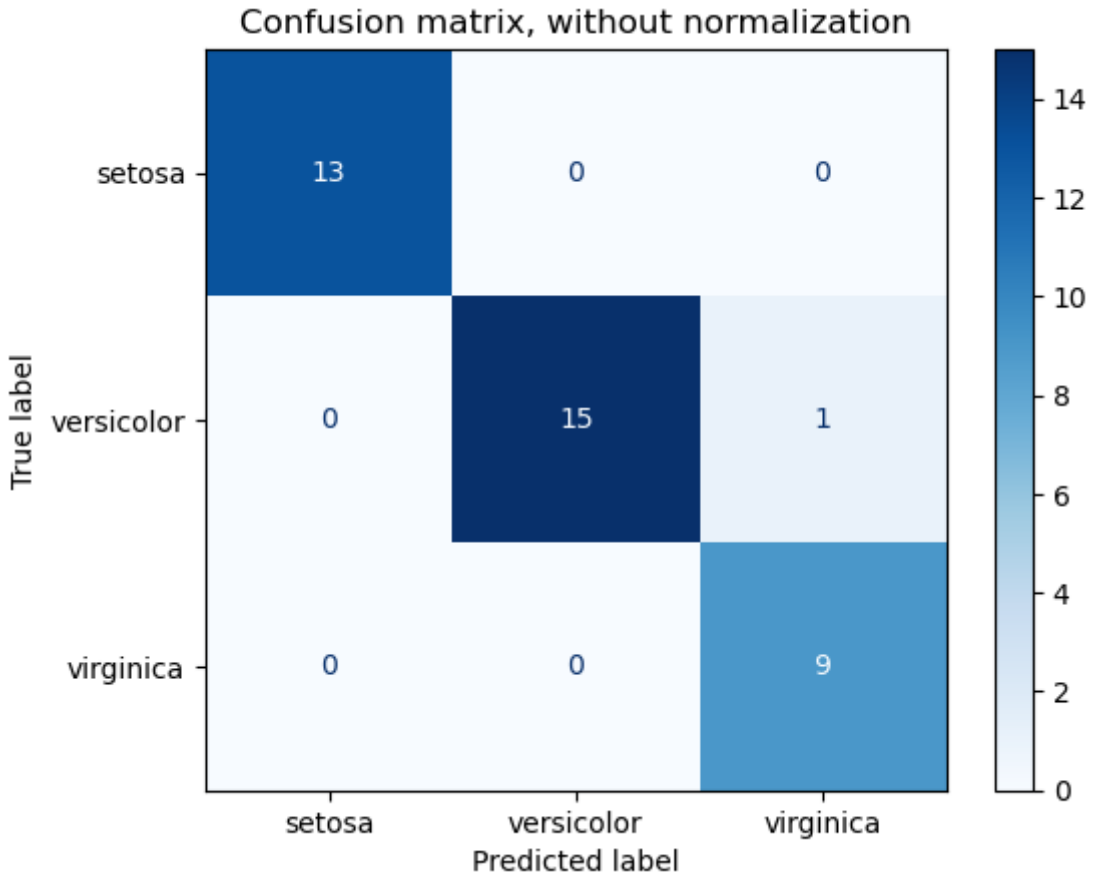


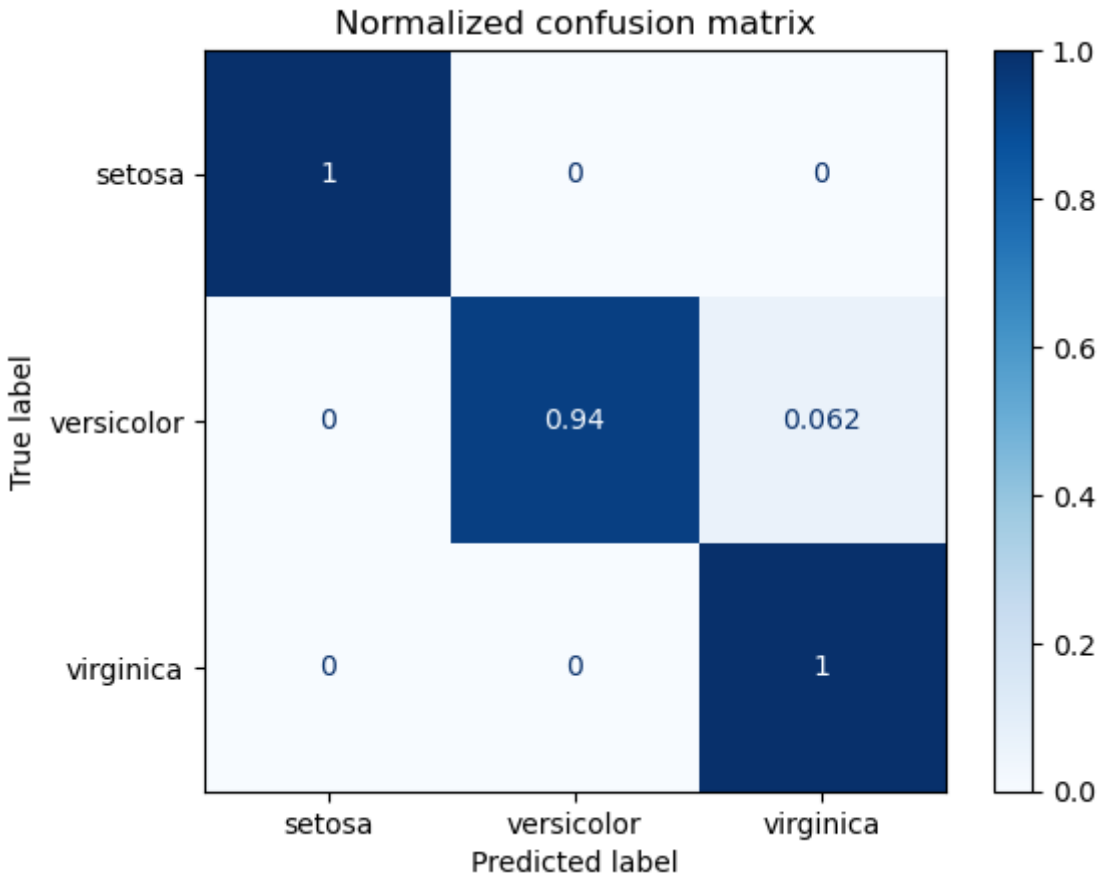
2b)

Using logistic regression as a classifier works well as shown in the confusion matrix. Only one point is mispredicted.

In [23]:
```
iris = load_iris()

#Loading the data the required data.
X = iris.data
y = iris.target
names = iris.target_names

#Split data into train sets and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

#Create the logistic regression classifier and train the model with the training da
logReg = LogisticRegression(C=0.5)
logReg.fit(X_train, y_train)

np.set_printoptions(precision=2)

#Creating the confusion matrices, one without normalization and one normalized
titles_options = [
    ("Confusion matrix, without normalization", None),
    ("Normalized confusion matrix", "true"),
    ]
for title, normalize in titles_options:
```

```
    disp = ConfusionMatrixDisplay.from_estimator(
        logReg,
        X_test,
        y_test,
        display_labels=names,
        cmap=plt.cm.Blues,
        normalize=normalize,
        )
    disp.ax_.set_title(title)

#Show plot
plt.show()
```





2c)

The weight of the nearest neightbours is split into two cases; distance and uniform.

When the weight is uniform there seems to be happening more mispredictions the higher k becomes, based on the data.

When the weight is distance there is less mispredictions the higher k is, altough there is no improvements after k =15.

For both of the cases, the areas for the different species become more distinct the higher k gets.

Because the species Versicolor and Virginica only had small differences, as shown in the visualization above, it is important that the weight is based on distance. Uniform weight performed decently when k=1, but this is because noise have a greater influence when k is small.

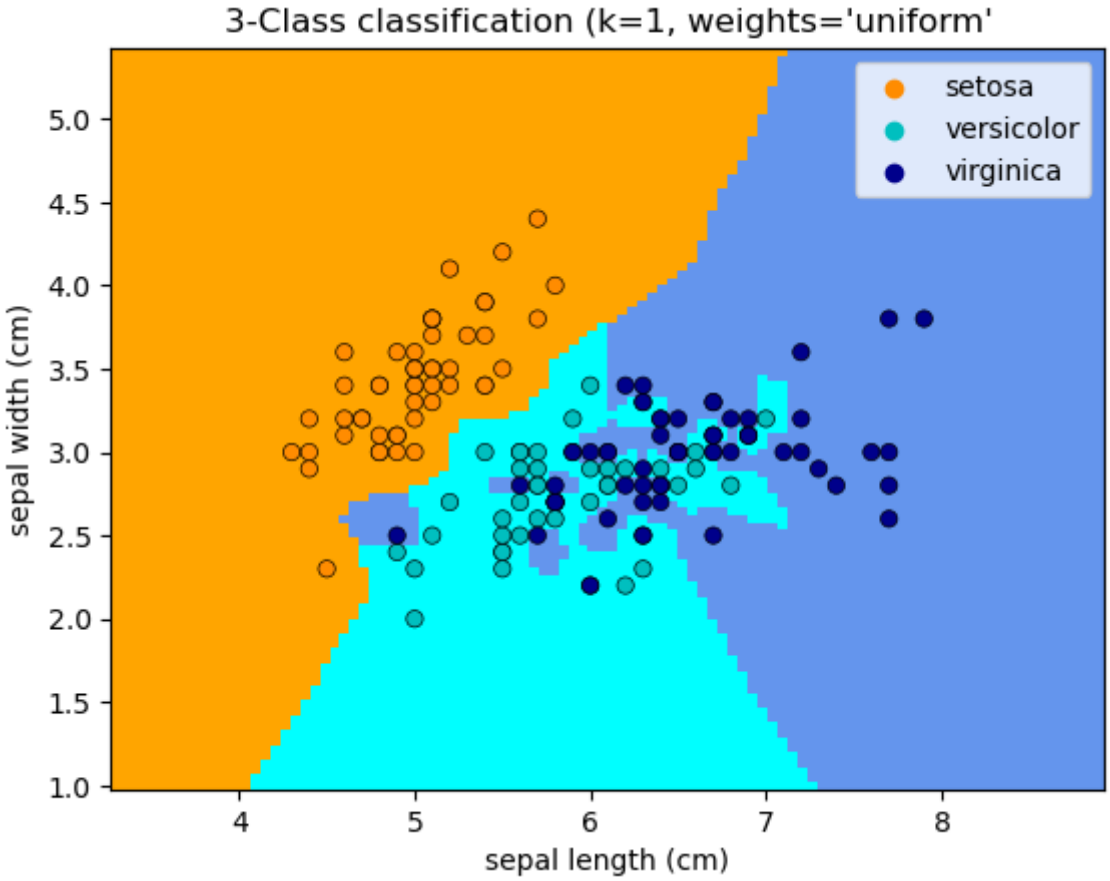
When the weight is based on distance a higher k gives more entities to take into account when classifiying. Even though some points might give incorrect information about what type of specie the entity is - having the weight being based on distant makes these less influential. To compare this with when weight is uniform, if you can compare with more neighbours then a lot of the points will be further away and consequently will not be of the correct specie. Because all of the neightbours now is considered to have the same weight there will more mispredictions.

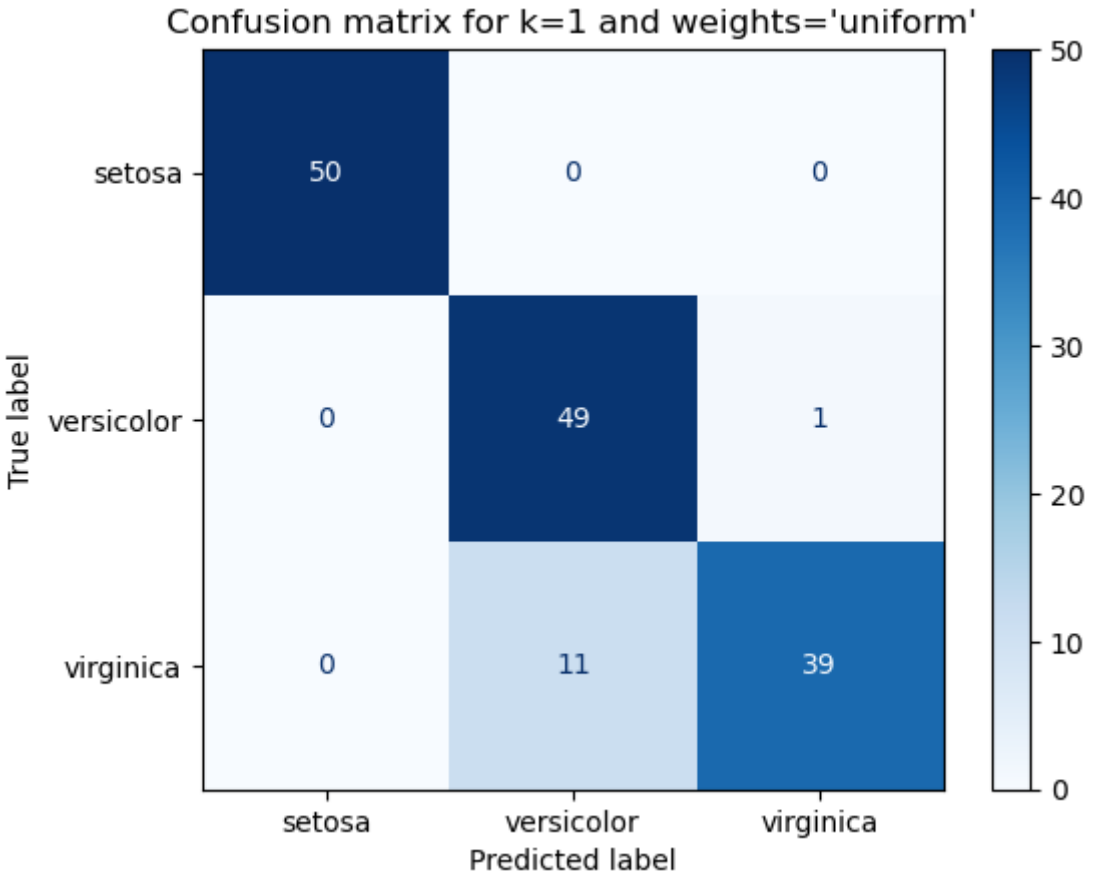
The best k for weight based on distance is: 29 and for uniform weight: 1.

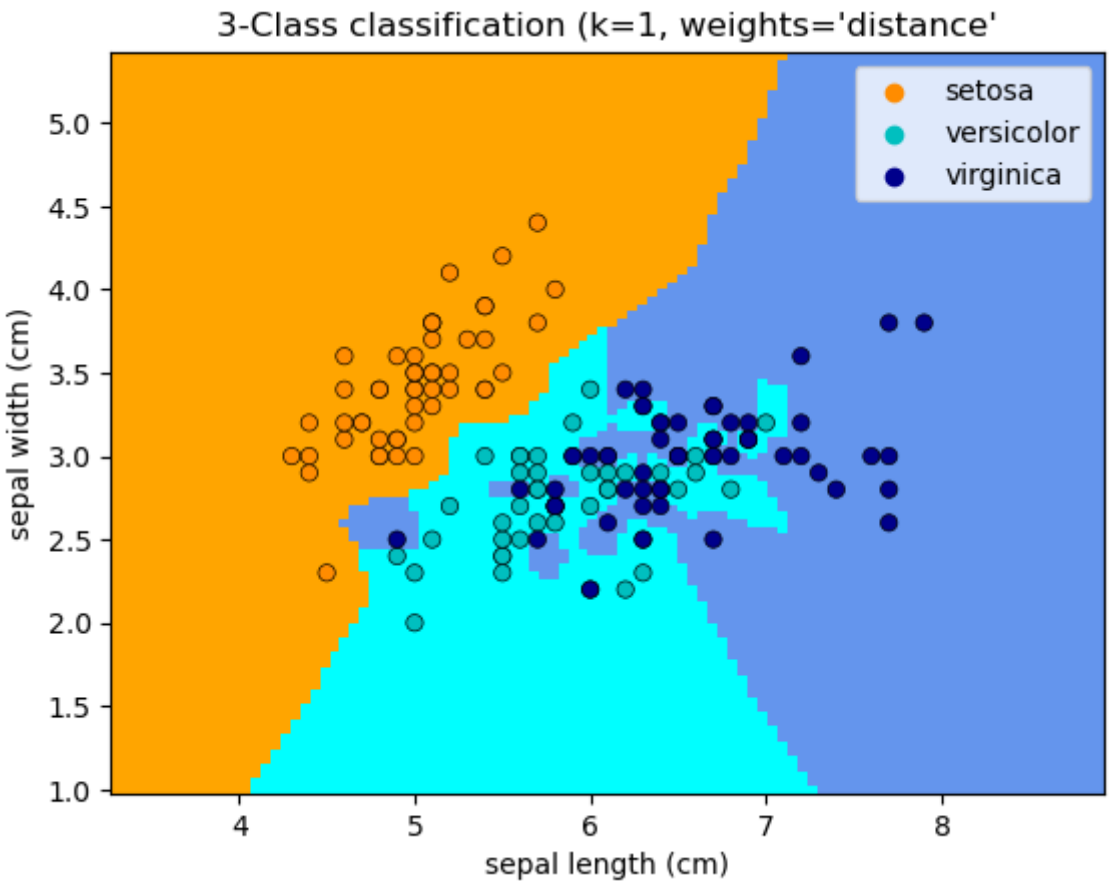
In [24]:

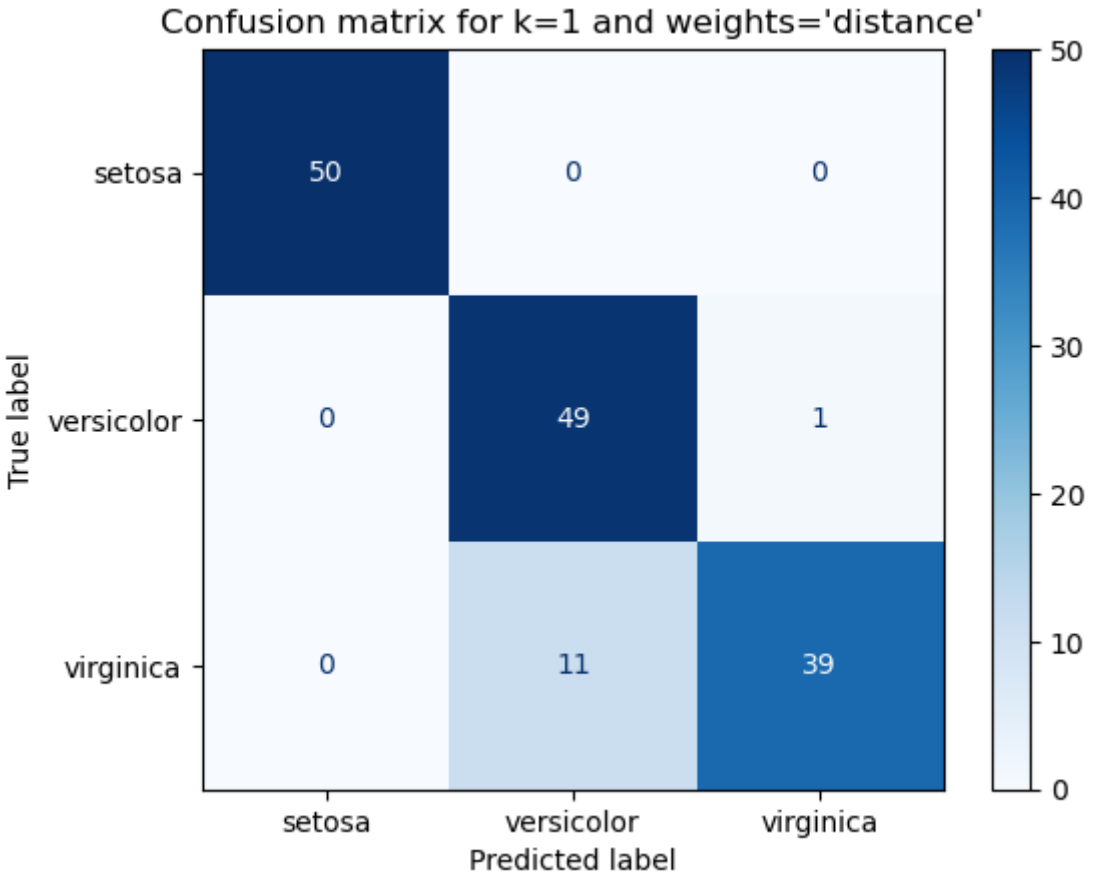
```
#The values for k - (1, 15, 29)
for n in range(1,30, 14):

    #Loading dataset
    iris = load_iris()

    #Loading data, only looking at the two first columns
    X = iris.data[:, :2]
    y = iris.target

    names = iris.target_names

    #Colors for plot
    cmap_light = ListedColormap(["orange", "cyan", "cornflowerblue"])
    cmap_bold = ["darkorange", "c", "darkblue"]

    #Creating the plots for the classifyers by scattering the data and drawing the
    #1. Uniform, 2. Distance
    for weights in ["uniform", "distance"]:
        #Create classifier and put in data
        clf = neighbors.KNeighborsClassifier(n, weights=weights)
        clf.fit(X, y)

        #Creating the visual display of the plot
        _, ax = plt.subplots()
        DecisionBoundaryDisplay.from_estimator(
            clf,
            X,
            cmap=cmap_light,
            ax=ax,
            response_method="predict",
            plot_method="pcolormesh",
            xlabel=iris.feature_names[0],
            ylabel=iris.feature_names[1],
            shading="auto"
        )

        #Scattering the data and coloring the respective specie
        sns.scatterplot(
            x=X[:, 0],
            y=X[:, 1],
            hue=iris.target_names[y],
            palette=cmap_bold,
            alpha=1.0,
            edgecolor="black"
        )
        #Creating title of the plot
        plt.title(
            "3-Class classification (k=%i, weights='%s'" % (n, weights)
        )
        #Creating the confusion matrix
        disp = ConfusionMatrixDisplay.from_estimator(
            clf,
            X,
            y,
            display_labels=names,
            cmap=plt.cm.Blues,
            normalize=None,
        )
        #Title for the confusion matrix
        disp.ax_.set_title("Confusion matrix for k=%i and weights='%s'" %(n, weigh
```

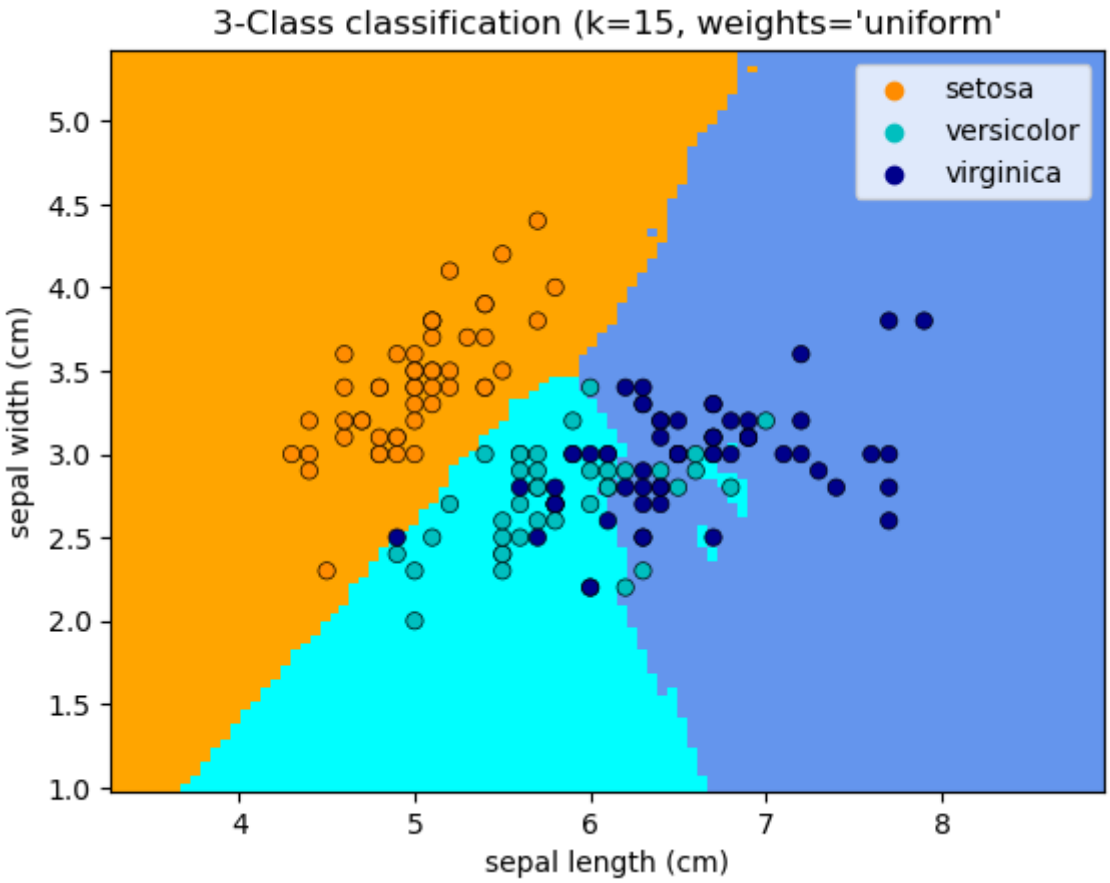
```
    #show plot
    plt.show()
```

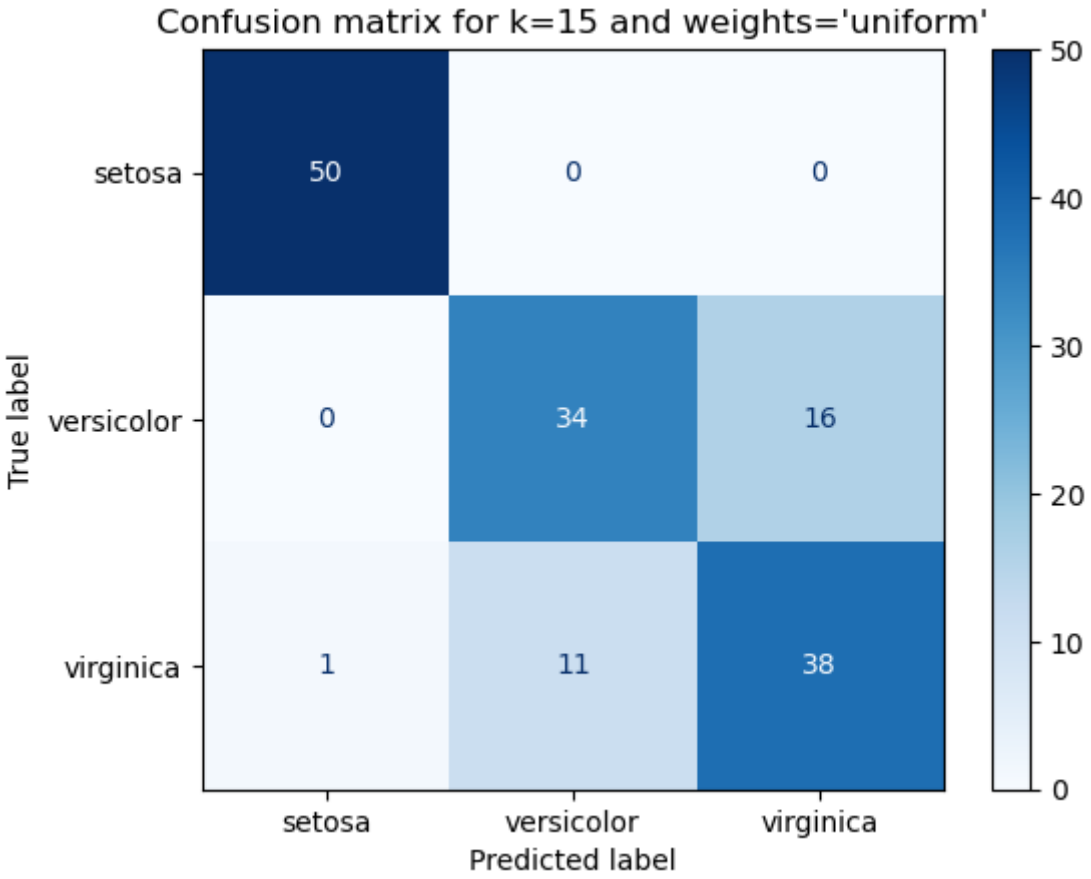


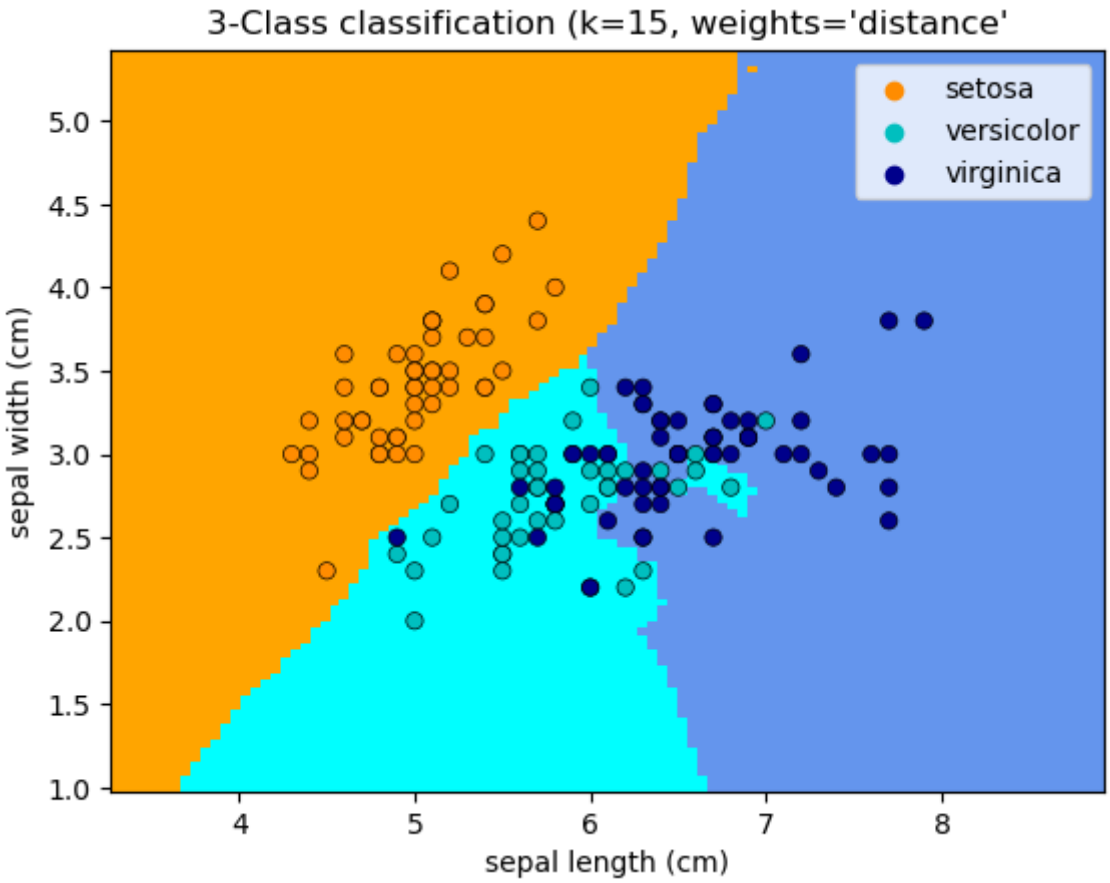


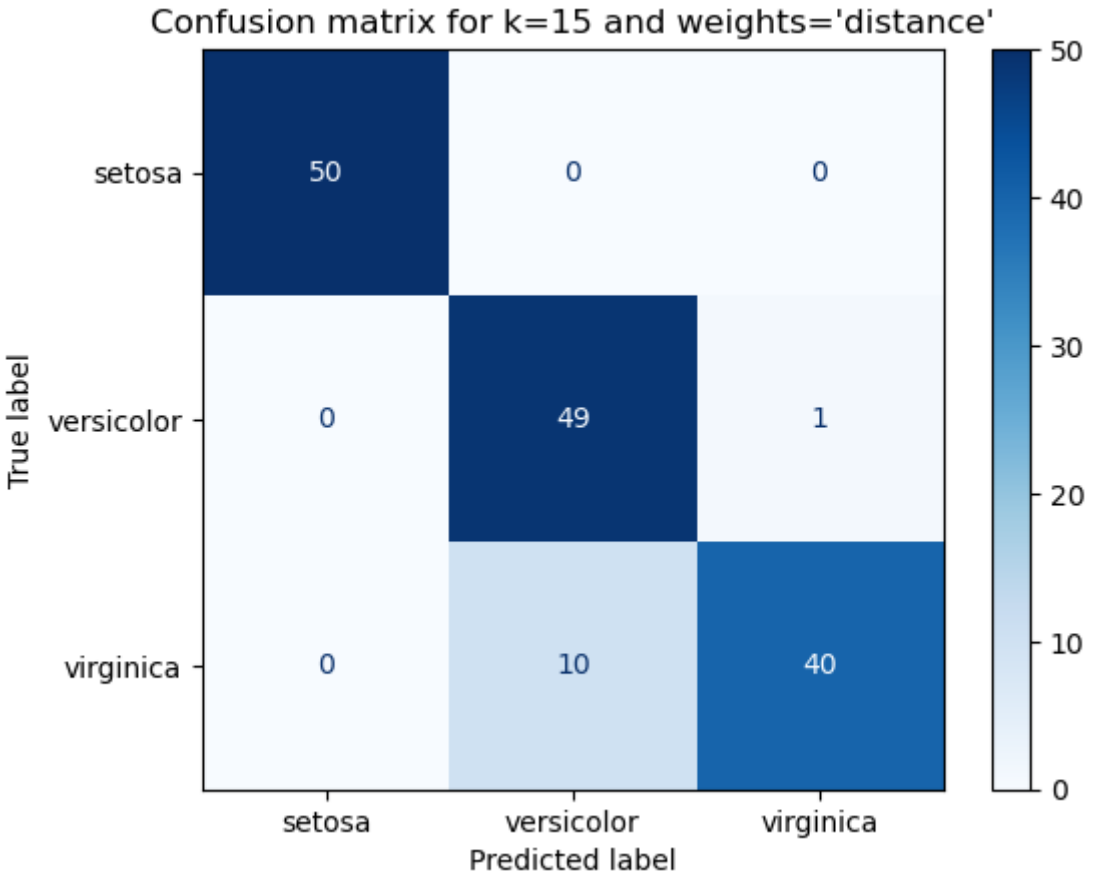

3-Class classification (k=1, weights='distance'
setosa
versicolor
virginica
sepal width (cm)
sepal length (cm)
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
4
5
6
7
8

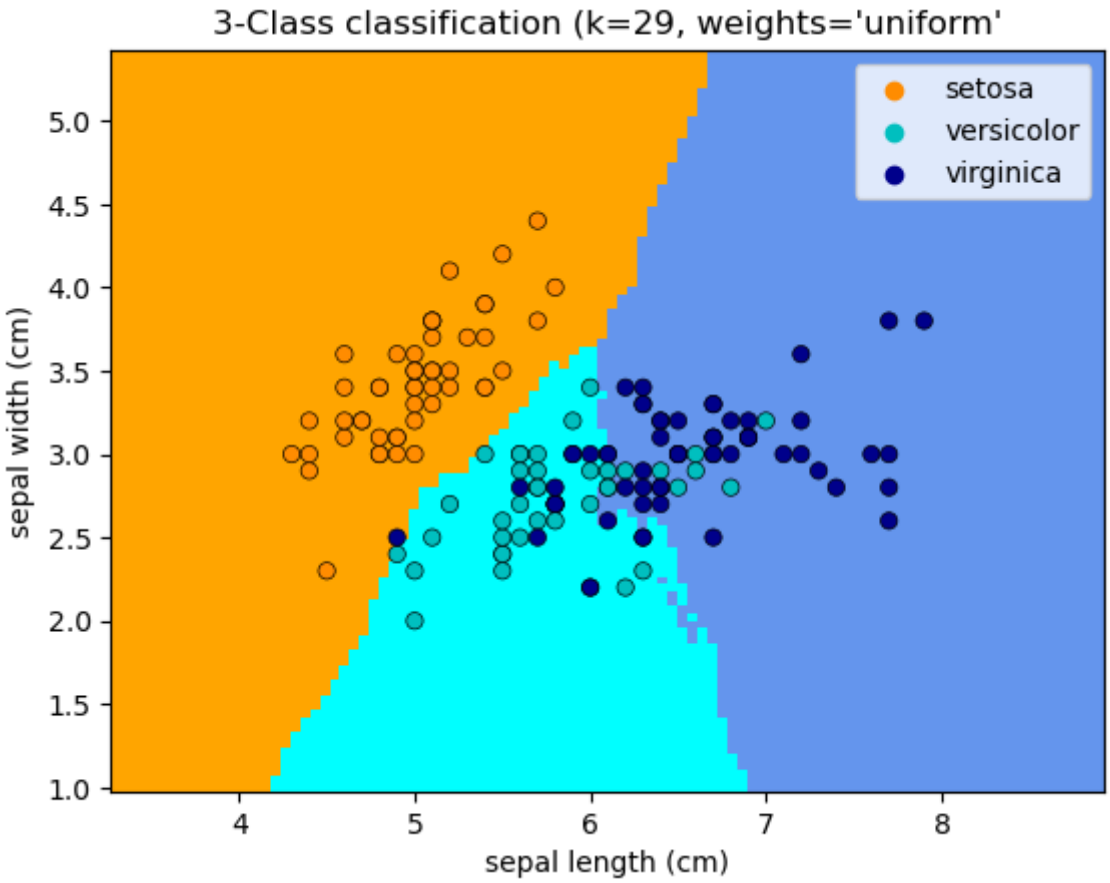

Confusion matrix for k=1 and weights='distance'
setosa
versicolor
virginica
True label
50
0
0
0
49
1
0
11
39
setosa
versicolor
virginica
Predicted label
50
40
30
20
10
0

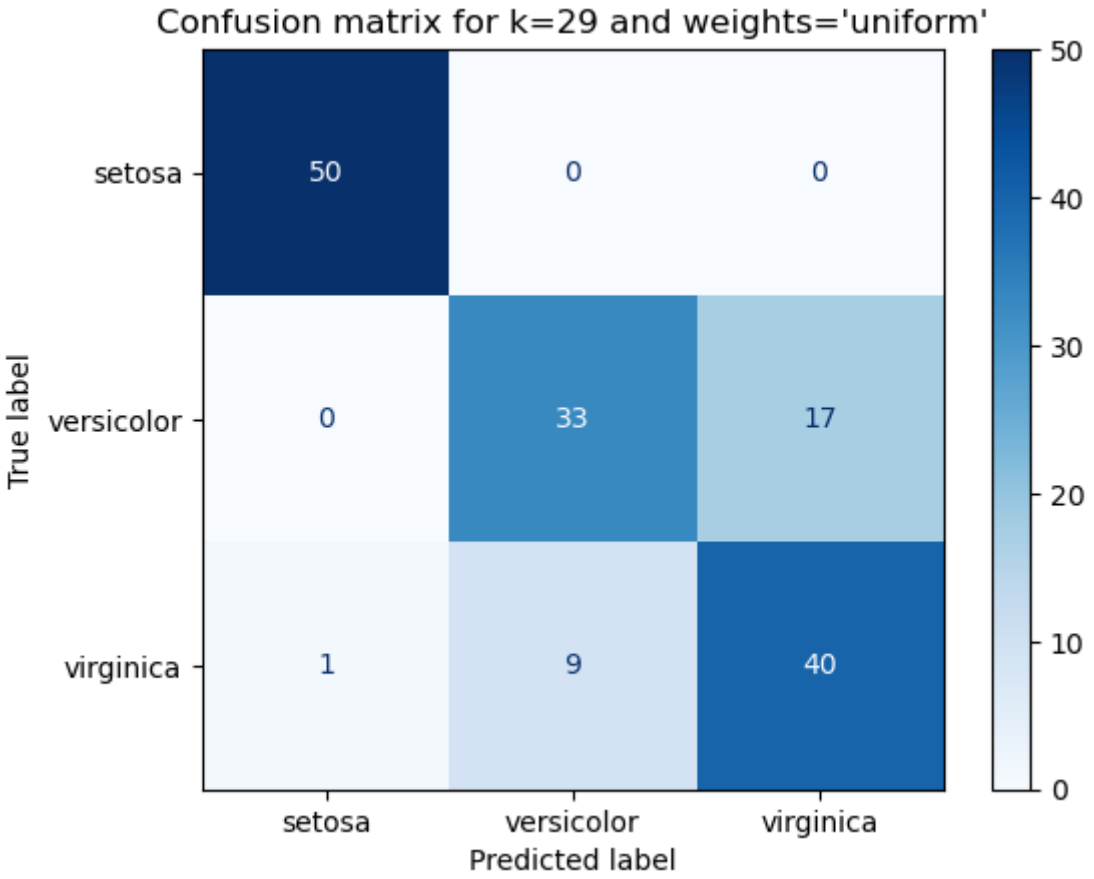

3-Class classification (k=15, weights='uniform'
setosa
versicolor
virginica
sepal width (cm)
sepal length (cm)
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
4
5
6
7
8

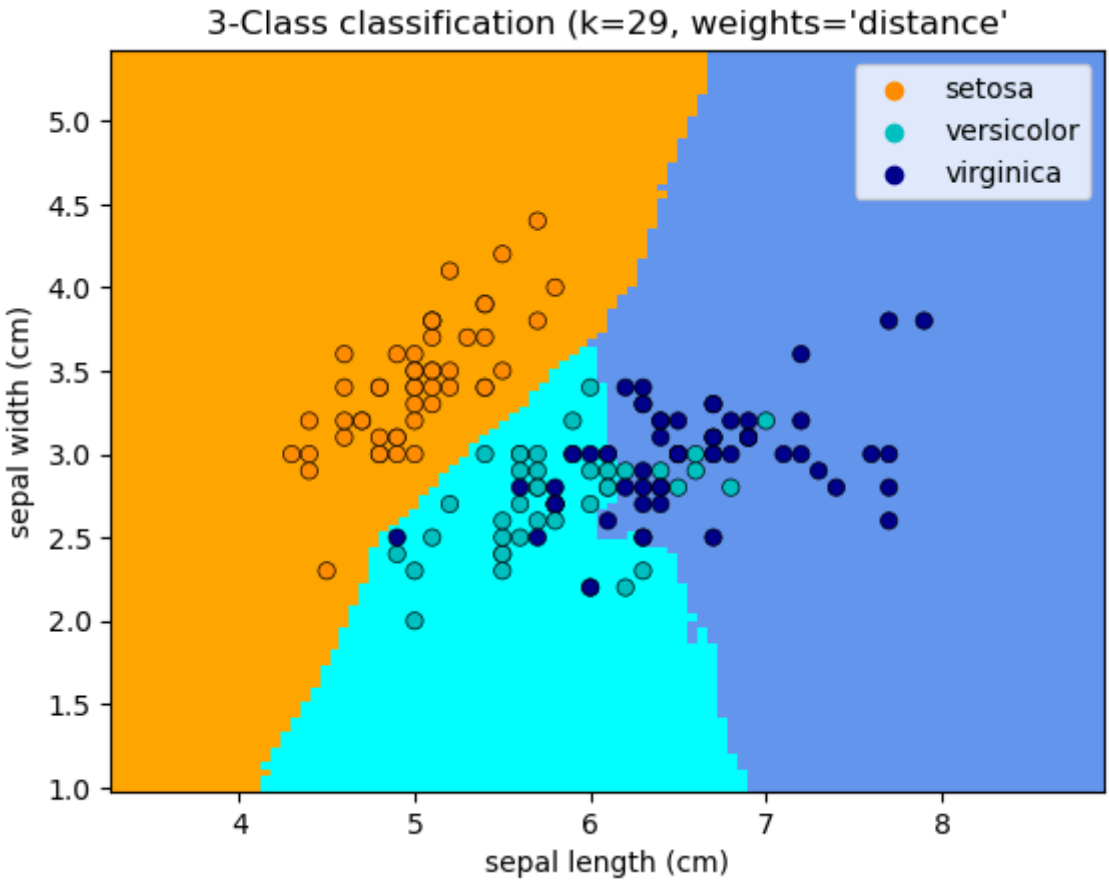

Confusion matrix for k=15 and weights='uniform'
True label
Predicted label
setosa
versicolor
virginica
50
0
0
0
34
16
1
11
38
50
40
30
20
10
0

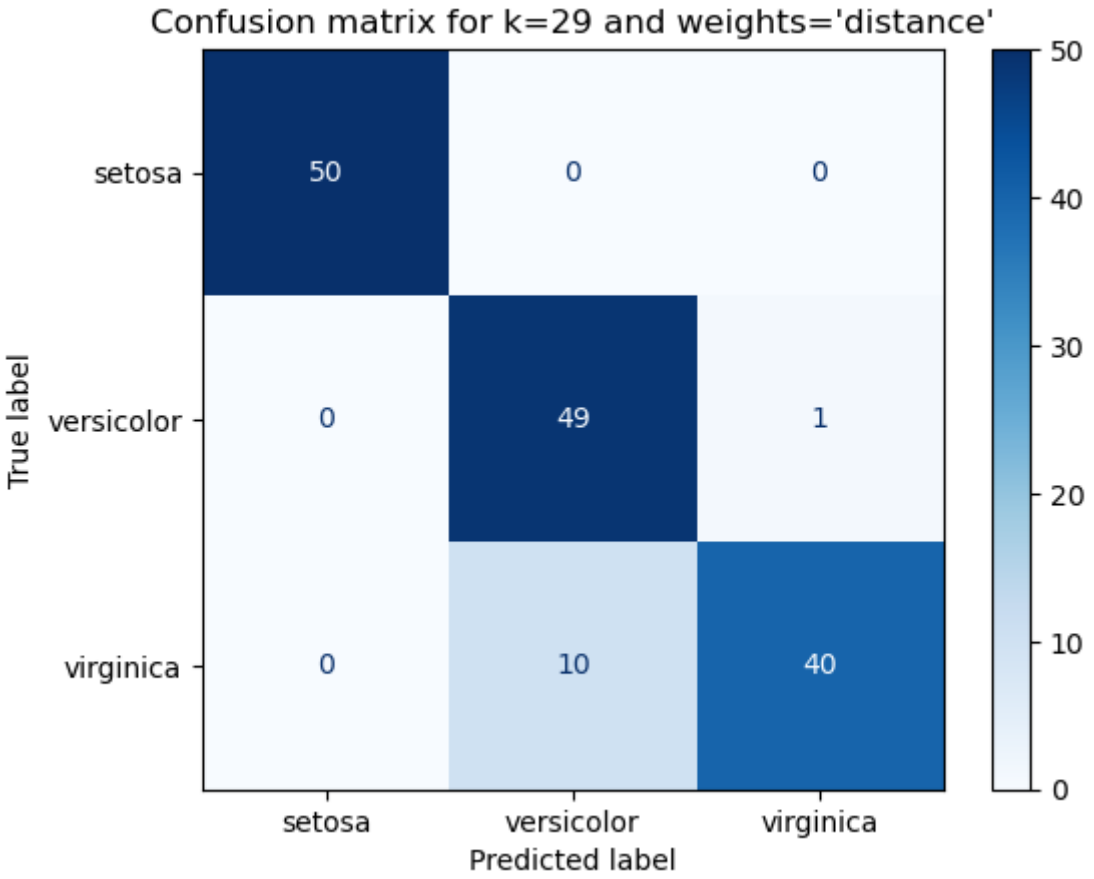

3-Class classification (k=15, weights='distance'
setosa
versicolor
virginica
sepal width (cm)
sepal length (cm)
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
4
5
6
7
8


Confusion matrix for k=15 and weights='distance'
setosa
versicolor
virginica
True label
50
0
0
0
49
1
0
10
40
setosa
versicolor
virginica
Predicted label
50
40
30
20
10
0


3-Class classification (k=29, weights='uniform'
setosa
versicolor
virginica
sepal width (cm)
sepal length (cm)
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
4
5
6
7
8


Confusion matrix for k=29 and weights='uniform'
setosa
versicolor
virginica
True label
50
0
0
0
33
17
1
9
40
setosa
versicolor
virginica
Predicted label
50
40
30
20
10
0





2d)

The logistic regression classifier does a greater job at predicting what class an entity is, which can be seen in the confusion matrix. The logistic regression model only mispredicts 1

out of 38 and the k-nearest neighbours mispredict 11 out of 150 at best.

Both of the models only struggles with predicting versicolor and virginica. The k-nearest neighbour biggest problem is wrongly classifying virginica as versicolor.

References

[1] Hemnet (2020). Retrieved from https://chalmers.instructure.com/courses/23445/assignments/70460?module_item_id=364234 [Online resource]

[2] R. A. Fisher (1936). "The use of multiple measurements in taxonomic problems". Imported from sklearn into python.