Project: Milestone 1

Chirayush Patel, Anand Ravishankar

Student ID: 114950240, 115075810

1 Classification Algorithm: Naive Bayes Classifier

The aim of this classifier is to determine the survival status of new data-points, given information of prior training data-sets.

1.1 Data description

The titanic data-set consists of 891 data instances and 13 features (+1 predicate feature). Each row represents one person. The feature description is provided below:

- survival - Survival
- PassengerID - Passenger ID
- Pclass - Passenger Class
- title - Title
- name - Name
- sex - Sex
- age - Age
- sibsp - Number of Siblings/Spouses Aboard
- parch - Number of Parents/Children Aboard
- ticket - Ticket Number
- fare - Passenger Fare
- cabin - Cabin
- Family_size = Size of family
- embarked - Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)

1.2 Data Preprocessing

Before the Naive-Bayes model can be trained on this data-set, we first had to clean the data and ensure that the model can be fit properly on the data. Note: We dropped the following features: ”PassengerId”, ”Name”, ”Ticket”, ”Cabin”, since these should not have any impact on the prediction process.

After gathering a description of the dataset, we encountered three main issues: two

- Categorical feature values: The features ”Embarked”, ”Title” and ”Sex” had categorical values and hence needed to be converted into numerical features. Each categorical feature was converted by a simple replacement operation. Example: Male was converted to 0 and ”Female” was converted to 1.
- Binning: The features ”Age” and ”Fare” had a large had to binned for having discrete values based on range.

1.3 Data Visualization

In this section we describe various plots related to the cleaned data-set. Figure 1 represents the prior distribution of the ”Survived” label. The next four plots are histograms representing the following (top-left, clockwise): 1. Plot of the survival probability for male and female passengers belonging to different classes. 2. The total count of passengers who survived and who didn't distributed based on different classes. 3. The total count of passengers who survived and who didn't distributed based on different gender. 4. The total count of passengers who survived and who didn't distributed based on different port of embarkment. Figure 2 shows the exhaustive pair-plots for all features.

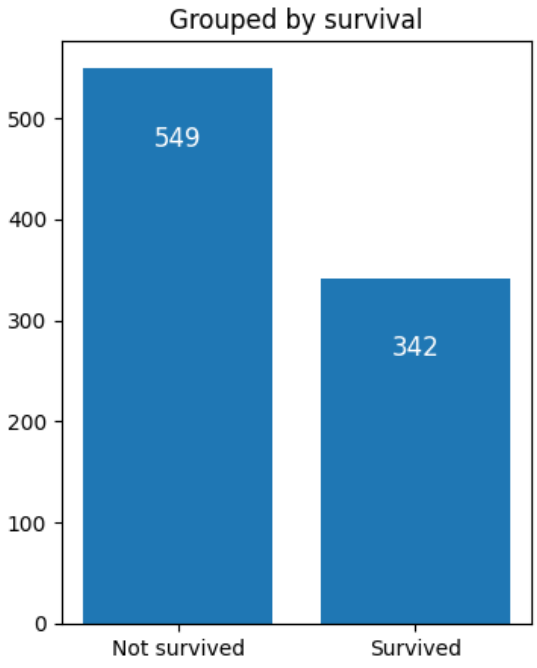
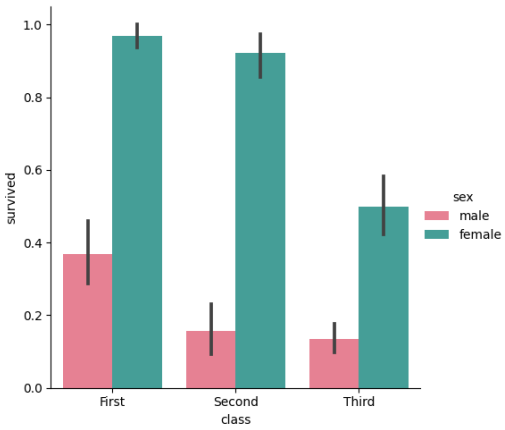
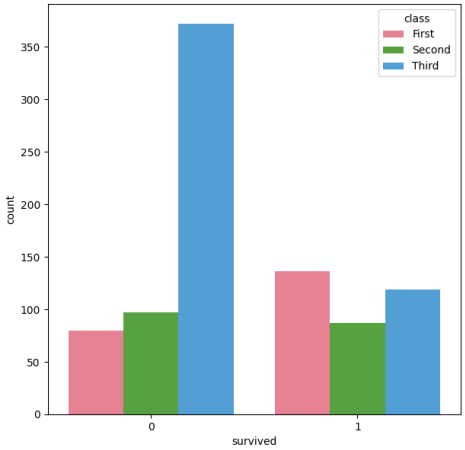
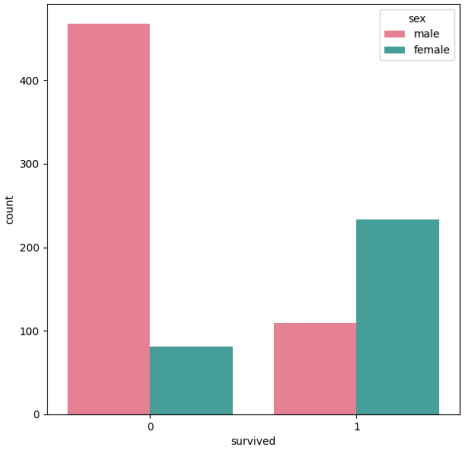
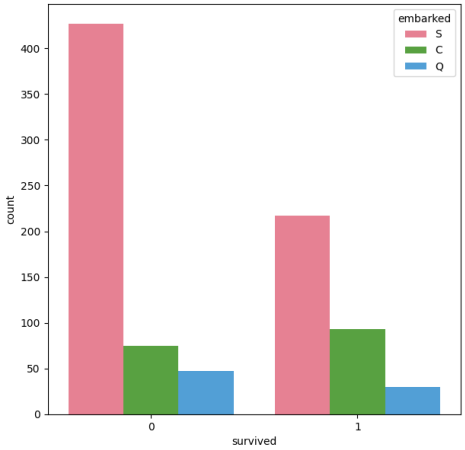


Figure 1: Prioris


survived
class
First
Second
Third
sex
male
female


count
survived
class
First
Second
Third


count
survived
sex
male
female


count
survived
embarked
S
C
Q

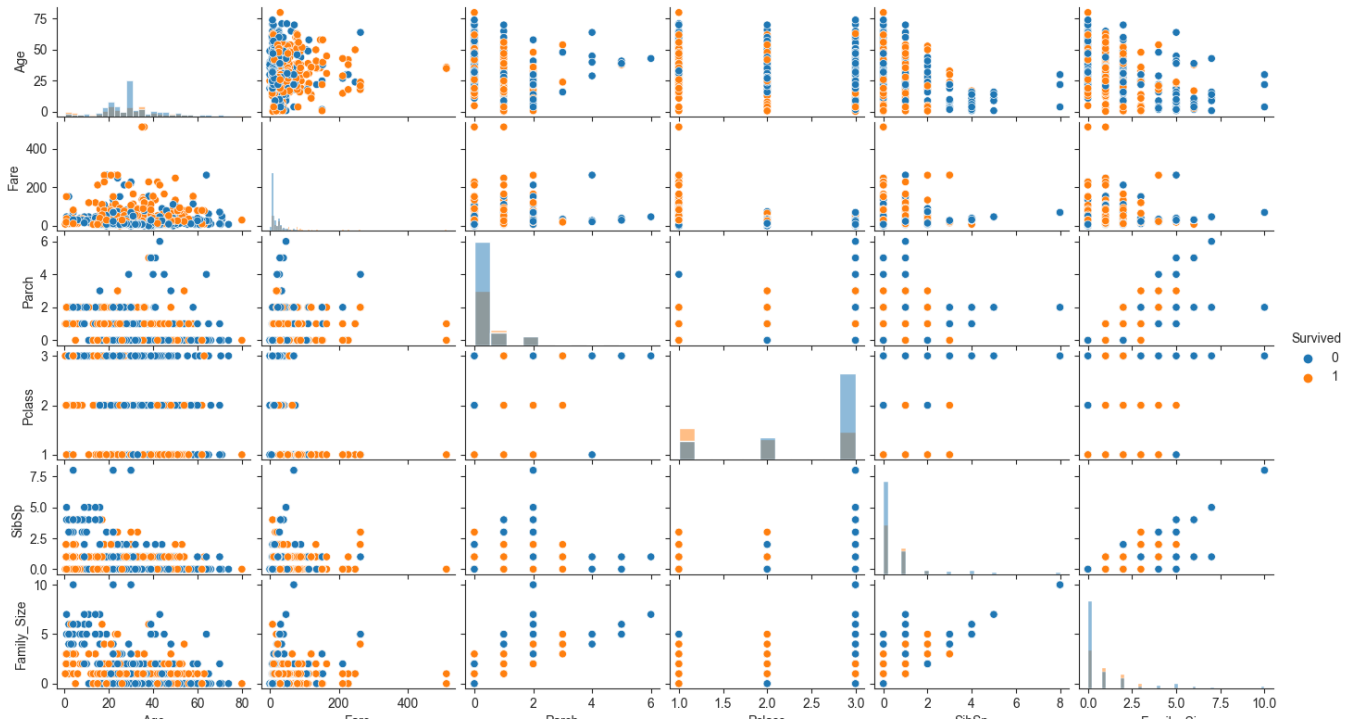


Figure 2: Pairplot

1.4 Naive Bayes Applied

After pre-processing, the data was split into a train and test set at a ratio of 80:20. The training set was further divided into data-frames where the target label was 0 or 1. The prioris were calculated from the number of 0 and 1 entries in the training set.

```
train, test = train_test_split(df, test_size=0.2)

survived_yes=train.loc[train.Survived==1]
survived_no=train.loc[train.Survived==0]

prior_yes=len(survived_yes)/len(train)
prior_no=len(survived_no)/len(train)
```

After the two prioris were calculated, we got the training subset for each individual feature and it's cardinality. Iterating over l features, the likelihood i.e. $P(data \mid label)$, where data = $x_{(i\,=\,1,\ldots,\,n)}$, was calculated as:

$$P(data|label) = \prod_{i=1}^{l} P(x_i|label)$$

```
for i in range(l):
    likelihood_yes = train[(train[atr[i]] == test1[i])
    & (train.Survived == 1)].count().values.item(0)
    //count of matching features with survival = 1
    py = py * (likelihood_yes) / len(survived_yes)
```

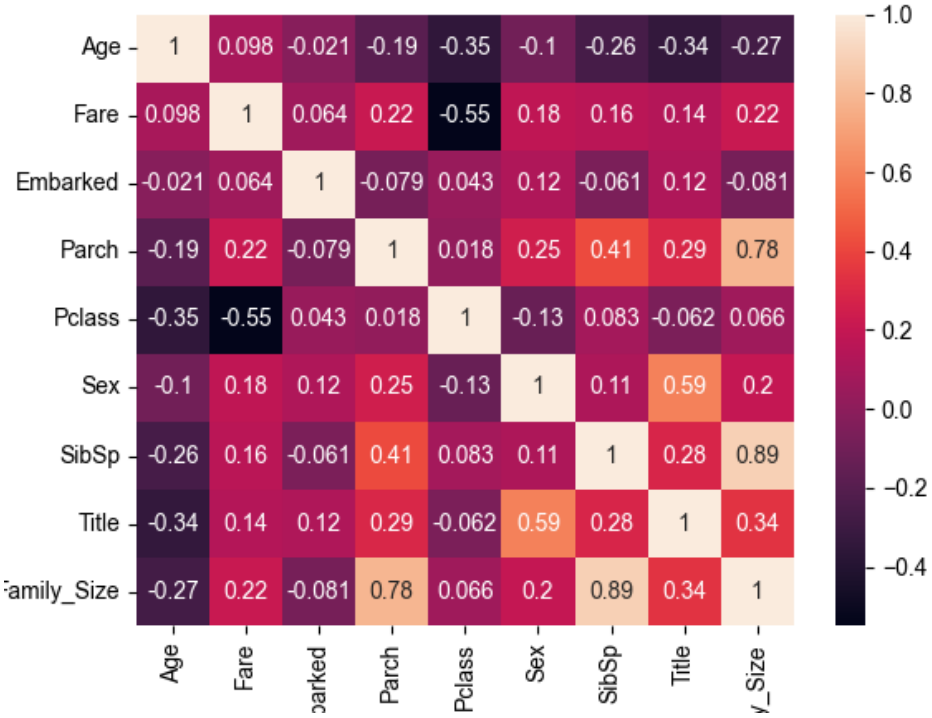


Figure 3: Correlation matrix with all features

	precision	recall	f1-score	support
0	0.69	0.86	0.77	99
1	0.75	0.53	0.62	80
accuracy			0.71	179
macro avg	0.72	0.69	0.69	179
weighted avg	0.72	0.71	0.70	179

Figure 4: Evaluation metrics with all the features considered

```
//running product of likelihood
total_yes = py * prior_yes //posterior probability
```

The final conditional probability was calculated by multiplying the likelihood with prioris. For each of the entry in the test set, the posterior was calculated and the calculated output was set to be 0 if posterior (label = 0) was greater than posterior (label = 1), and vice versa. Two lists were maintained for the ground truth and the predicted value.

1.5 Effect of High correlation

The features ”Age” and ”Fare” had extreme high correlation with other features, as shown in Figure 3. The evaluation metrics for Naive Bayes with all the above shown features are shown in Figure 4. As we can see, the model performs decently well. If we remove the correlated features we should see a marked increase in the performance of our model. This is validated by the metrics shown in Figure 6. However

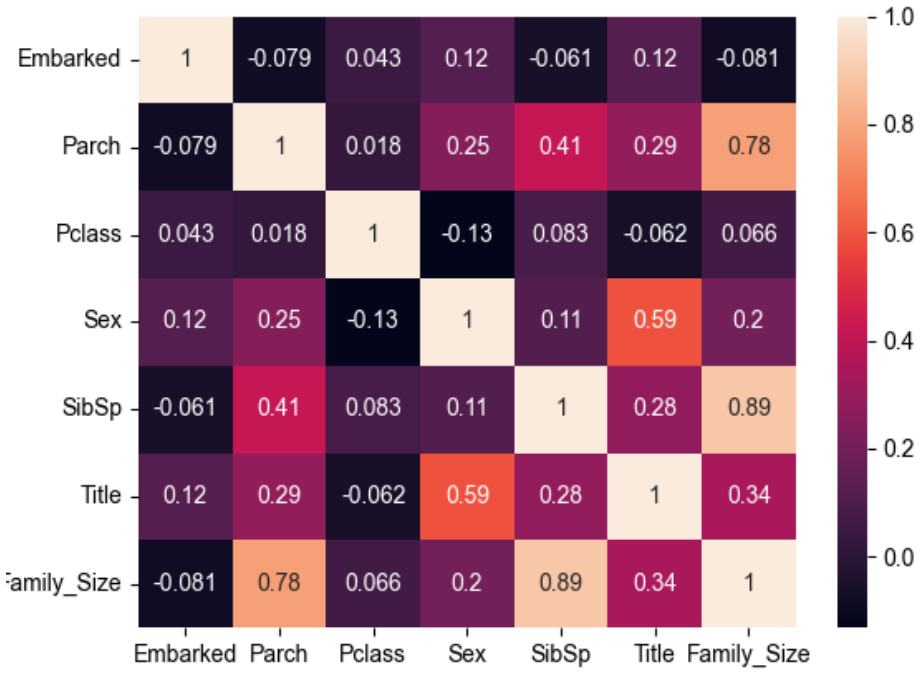


Figure 5: Correlation matrix with all features, except Age and Fare

	precision	recall	f1-score	support
0	0.83	0.84	0.83	99
1	0.80	0.79	0.79	80
accuracy			0.82	179
macro avg	0.81	0.81	0.81	179
weighted avg	0.82	0.82	0.82	179

Figure 6: Evaluation metrics with some of the features considered

it should also be noted that there is scope for improvement, as evidenced by the fact that there are still highly correlated features 5.

2 Clustering Algorithm: K-means

K-Means is a clustering algorithm which we have implemented using sklearn.KMeans library. The target dataset is the Iris dataset, which the target label discarded for clustering purposes. Towards the end we compare how well Kmeans has got to the ground truth by re-introducing the true labels.

2.1 Data Description

The data used for clustering was the Iris dataset. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant: Setosa, Versicolour, Virginica.

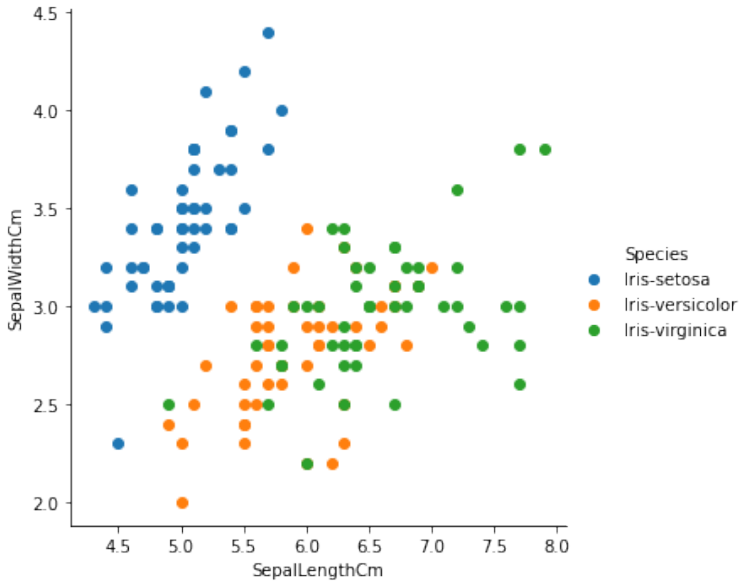


Figure 7: Scatter Plot of all three species along Sepal width and length

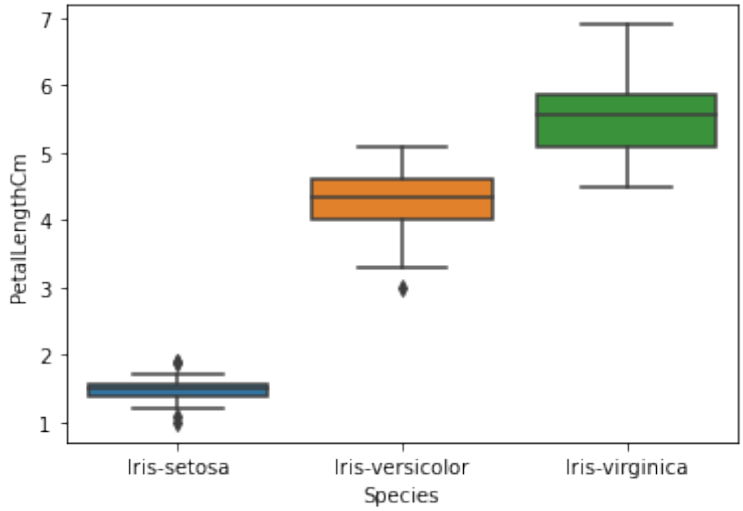


Figure 8: Box-and-whisker plot of all data-points for the petal length feature

No data pre-processing was required since the data was numerical and had low variance, with no missing values.

2.2 Data Visualization

In this section, we describe the various plots related to the Iris dataset. The descriptions for each plot have been provided as the caption for images 7 to 10. It is evident from the plots that Iris-Versicolour and Iris-Virginica will be difficult to differentiate since for all the features they are scattered almost always together.

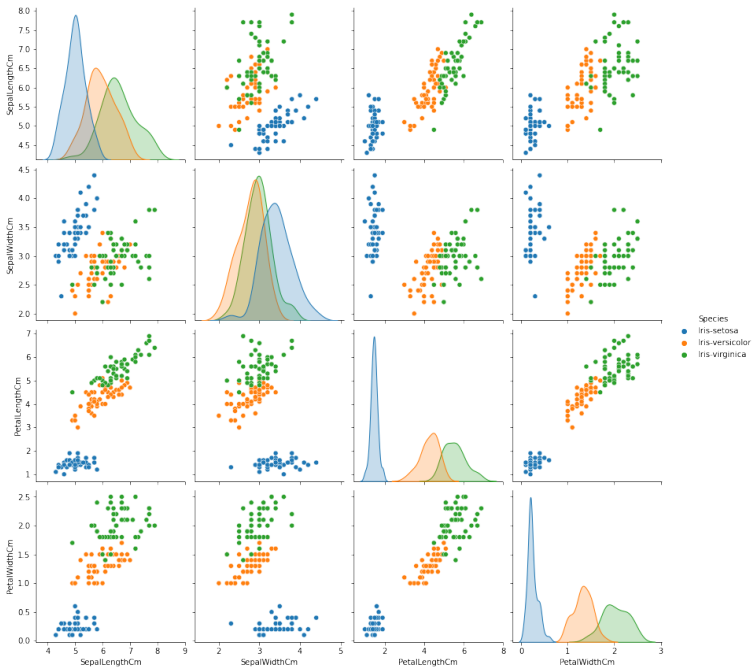


Figure 9: Pairplot to analyze the relationship between species for all characteristic combinations.

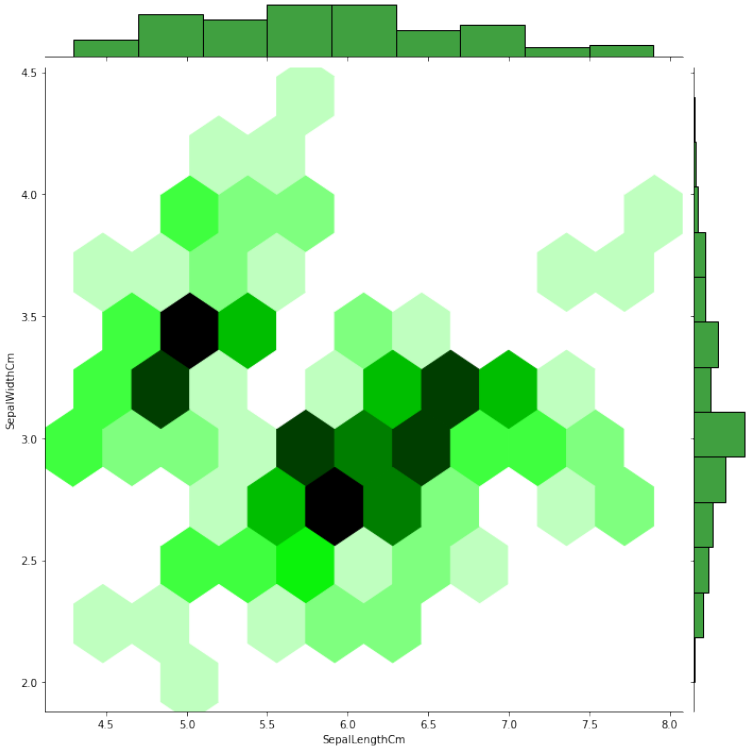


Figure 10: Hexagonal bin plot

2.3 Applying Kmeans

After loading the required libraries, we separated the features from the target label. To check the capabilities of the Kmeans model, we tested it on three different fronts.

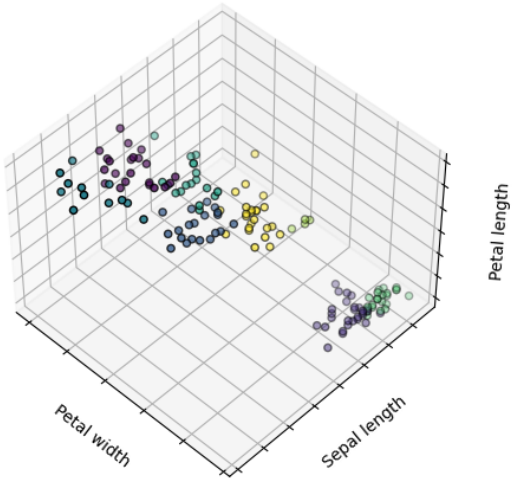


Figure 11: When forced with extra clusters, K-Means performs poorly leading to sub-optimal solution

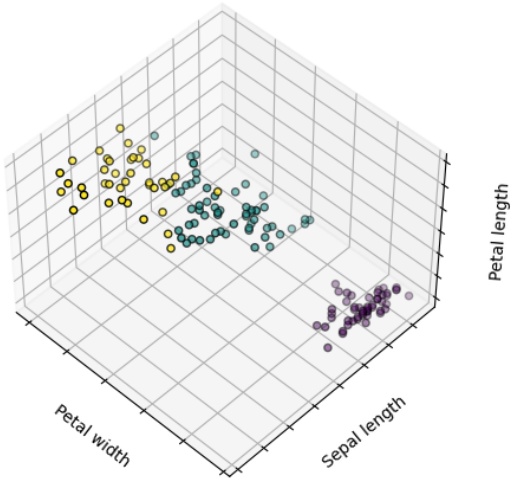


Figure 12: Random initialization accompanies highly volatile conditions and provides for a scenario where the selected centroids are not well positioned throughout the entire data space.

- Extra clusters: In this case we provided 3 extra clusters for the Kmeans to create as a parameter.
- Poor initialization: We demonstrate deviation from the ground truth even with optimal cluster, but poor initialization of centroids.
- Optimal number of cluster.

As we can deduce from our observations, for K-Means to perform well, it must be provided with an optimal number of clusters. Furthermore the centroid initialization method must not be random which could cause improper class labelling.

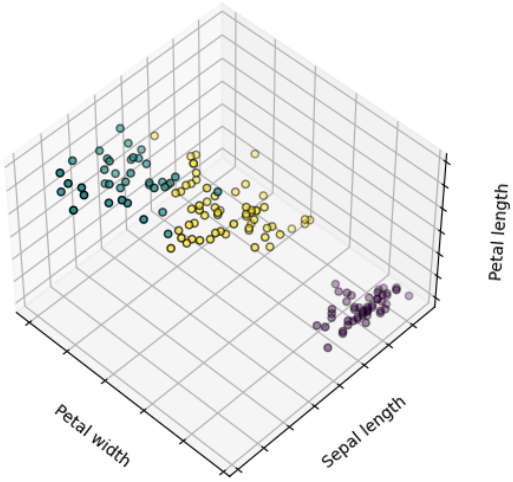


Figure 13: Optimal number of cluster along with k-means++ strategy of initialization

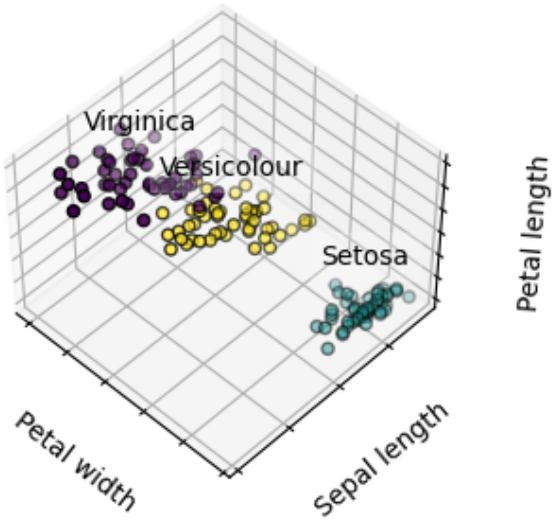


Figure 14: Ground truth