Naive Bayes Classification for Titanic Disaster

Final Thesis - Analytical Information Systems (SS2021)

at Hof University of Applied Sciences
Department of Computer Science
Software Engineering for Industrial Applications

Submitted to
Prof. Dr. Gefei Zhang
Alfons-Goppel-Platz 1
95028 Hof
Germany

Submitted by
Manoj Prabakar Kanagaraj
Fabrikzeile 26
92028 Hof
Germany

Hof, July 18, 2021

Table of Contents

List of figures

1 Problem Statement

Titanic disaster occurred 100 years ago on April 15, 1912, killing about 1500 passengers and crew members. The fateful incidents still compel the researchers and analysts to understand what could have led to the survival of some passengers and demise of the others. With the use of machine learning methods and a dataset consisting of 891 rows in the train set and 418 rows in the test set, I attempt to determine the correlation between factors such as age, sex, passenger class, to the chance of survival of the passengers. These factors may or may not have impacted the survival rates of the passengers. Here I have used naive bayes classification algorithm for predicting the survival rate in test data set.

2 Introduction

The field of machine learning has allowed analysts to uncover insights from historical data and past events. Titanic disaster is one of the most famous shipwrecks in the world history. Titanic is a British cruise liner that sank in the North Atlantic Ocean few hours after colliding with an iceberg While there are facts available to support the cause of the shipwreck, there are various speculations regarding the survival rate of passengers in the Titanic disaster. Over the years, data of survived as well as deceased passengers has been collected. The dataset has been studied and analysed using Naive Bayes Classification Algorithm.

3 Data set

The dataset found on the Kaggle website has two perspectives. One is a training data, and the other is a testing data. The objective of the training data is to create a model which will help in predicting the outcomes of the test data. The dataset I use for this classification was provided by the Kaggle website. The data consists of 891 rows in the train set which is a passenger sample with their associated labels. For each passenger, we were also provided with the name of the passenger, sex, age, his or her passenger class, number of siblings or spouse on board, number of parents or children aboard, cabin, ticket number, fare of the ticket and embarkation. The data is in the form of a CSV (Comma Separated Value) file. For the test data, we were given a sample of 418 passengers in the same CSV format.

4 State of the art

4.1 Technologies Used:

4.1.1 Classification Algorithm

Naive Bayes is a classification algorithm that applies Bayes theorem to build a prediction model.

$$P(Cj \mid D) = \frac{P(D \mid Cj) \times P(Cj)}{P(D)}$$

• P(Cj | D) = Probability of instance D being in class Cj , This is what we are trying to compute.

• P(D | Cj) = Probability of generating instance D given class Cj , We can imagine that being in class Cj , causes you to have feature D with some probability.

• P(Cj) = Probability of occurrence of class Cj, This is just how frequent the class Cj, is in our database.

• P(D) = Probability of instance D occurring. This can be ignored since it is the same for all classes.

Prediction involved calculating the probability for a particular data sample which belong to each of the two class. The class with the largest probability value was considered as the predicted class.

Bayes theorem use methodology of conditional probability.

4.1.2 Conditional Probability

Conditional probability is defined as the likelihood of an event or outcome occurring, based on the occurrence of a previous event or outcome. Conditional probability is calculated by multiplying the probability of the preceding event by the updated probability of the succeeding, or conditional, event.

The conditional probability can be calculated as:

P(A | B) = P(A, B) / P(B)

Firstly, in general, the result P(A | B) is referred to as the **posterior probability** and P(A) is referred to as the **prior probability** whereas P(B|A) is **likelihood** and P(B) is **evidence**.

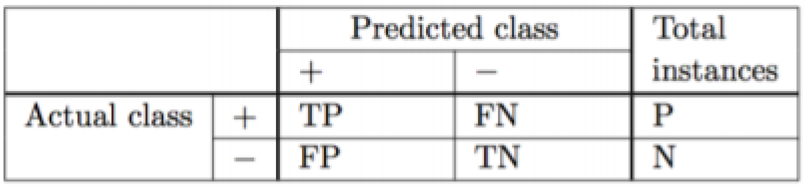
4.1.2.1 Example for conditional probability

When you flip a fair coin, there is an equal chance of getting either heads or tails. So, you can say the probability of getting heads is 50%.

Similarly, what would be the probability of getting a 1 when you roll a dice with 6 faces? Assuming the dice is fair, the probability of 1/6 = 0.166.

4.2 Performance Metrics

Accuracy performance metrics can be decisive when dealing with imbalanced data. The predictive accuracy, precision, recall, and F1 score gives better intuition of prediction results as compared to accuracy.



		Predicted class		Total instances
		+	−	
Actual class	+	TP	FN	P
	−	FP	TN	N

True Positive (TP) — model correctly predicts the positive class (prediction and actual both are positive).

True Negative (TN) — model correctly predicts the negative class (prediction and actual both are negative).
False Positive (FP) — model gives the wrong prediction of the negative class (predicted-positive, actual-negative).
False Negative (FN) — model wrongly predicts the positive class (predicted-negative, actual-positive).

4.2.1 Predictive Accuracy

Out of total values i.e., sum of total positive and total negative, what percentage are sum of truly positive and truly negative

$$\text{Predictive Accuracy} = (TP + TN) / (P + N)$$

4.2.2 Precision

Out of all the positive predicted, what percentage is truly positive.

$$\text{Precision} = TP / (TP + FP)$$

4.2.3 Recall (TP Rate)

Out of the total positive, what percentage are predicted positive

$$\text{Recall} = TP / (TP + FN) = TP/P$$

4.2.4 FP Rate

Out of the total negative, what percentage are predicted negative

$$\text{FP Rate} = FP / (FP + TN) = FP/N$$

4.2.5 F1-Score

It is the harmonic mean of precision and recall. It takes both false positive and false negatives into account. Therefore, it performs well on an imbalanced dataset.

$$\text{F1 Score} = (2 \times \text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$$

4.3 Tools Used

Here I have used **Jupyter Notebook** from the **Anaconda Navigator** to run my python code

5 Why Naive Bayes

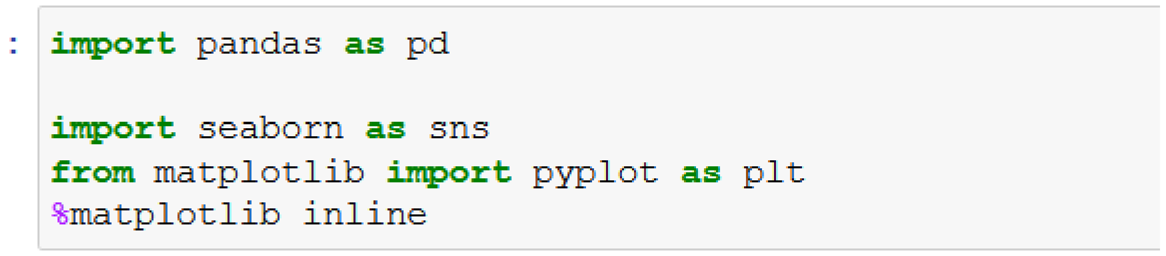
I chose this algorithm because **it is easy and fast to predict the class of the test data set**.

It also performs well in **multi-class prediction**.

When assumption of independence holds, a Naive Bayes classifier performs better compared to other models

6 Implementation

6.1 Import library



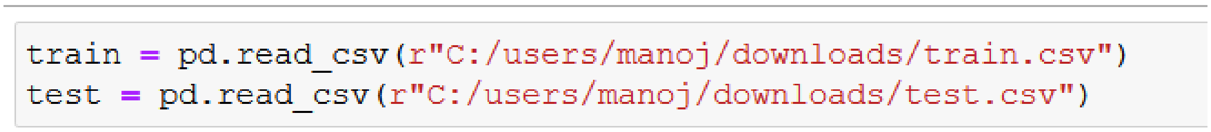
```
: import pandas as pd

  import seaborn as sns
  from matplotlib import pyplot as plt
  %matplotlib inline
```

Figure 1: Import Library

6.1.1 Pandas

It is the open source library for python language. Using Pandas we can import data from excel file to data frame. Pandas library is a very high-level Python library that provides high performance easy to use data structures. It has many functions for data importing, manipulation and analysis.

Using Pandas library I have created two data frames called **train** and **test** which holds the data of **train.csv** and **test.csv** respectively.



```
train = pd.read_csv(r"C:/users/manoj/downloads/train.csv")
test = pd.read_csv(r"C:/users/manoj/downloads/test.csv")
```

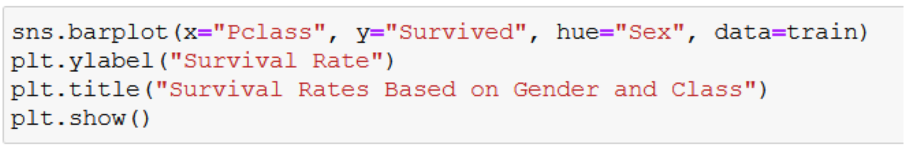
Figure 2: Read Excel File

6.1.2 Seaborn

It is the python data visualization library based on matplotlib. It is mainly used for statistical visualization. It provides a variety of visualization patterns. It uses fewer syntax and has easily interesting default themes.

6.1.3 Matplotlib

Matplotlib is mainly deployed for basic plotting. Visualization with matplotlib generally comprises of Bar chart, Pie chart, etc. It is popular package that provides 2D plotting, as well as 3D plotting.



```
sns.barplot(x="Pclass", y="Survived", hue="Sex", data=train)
plt.ylabel("Survival Rate")
plt.title("Survival Rates Based on Gender and Class")
plt.show()
```

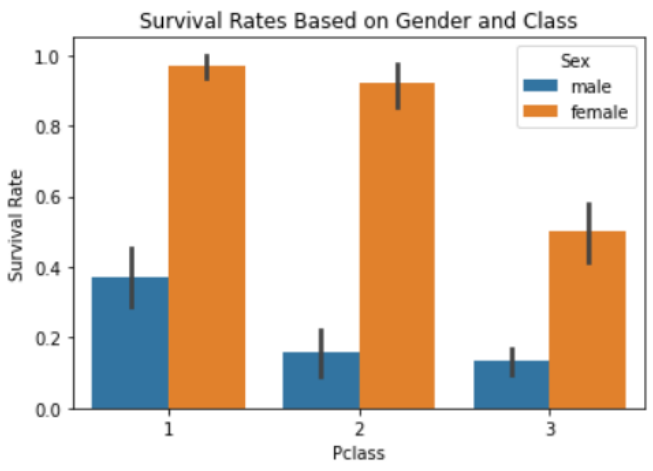


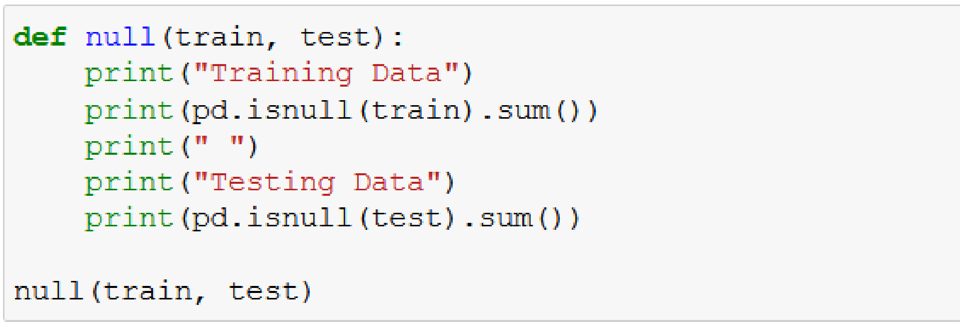
Figure 3: Data Visualization

6.2 Data cleaning

It is the process of removing null records, dropping unnecessary columns, treating missing values, restructuring the data to modify it to a more readable format.

6.2.1 Detecting null values

Detecting null values from the data sets and then replacing the null values with average value of that specified column.

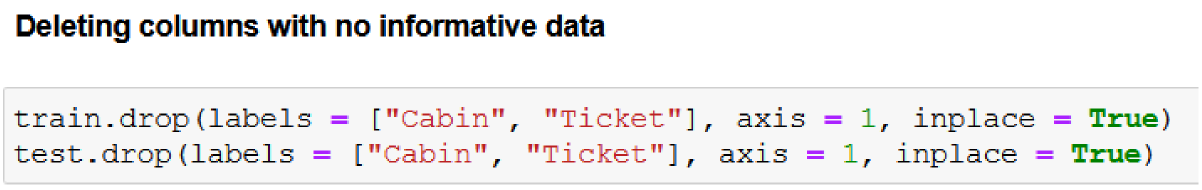


```
def null(train, test):
    print("Training Data")
    print(pd.isnull(train).sum())
    print(" ")
    print("Testing Data")
    print(pd.isnull(test).sum())

null(train, test)
```

Figure 4: Detecting null values

6.2.2 Removing columns

Removing the column with less informative data for building the model. This does not provide any effect on the dataset



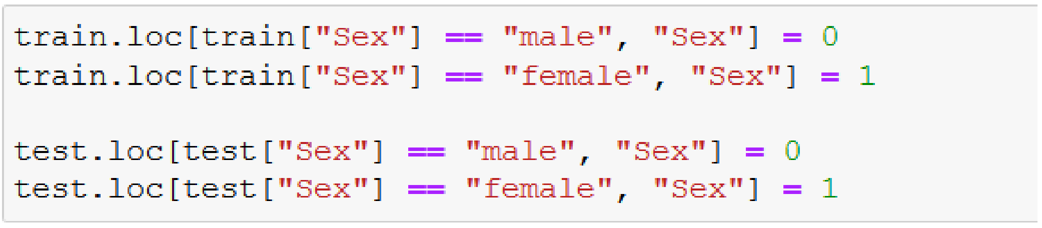
Deleting columns with no informative data

```
train.drop(labels = ["Cabin", "Ticket"], axis = 1, inplace = True)
test.drop(labels = ["Cabin", "Ticket"], axis = 1, inplace = True)
```

Figure 5: Remove Columns

6.2.3 Numerical values

Changing the categorical values into numerical one.



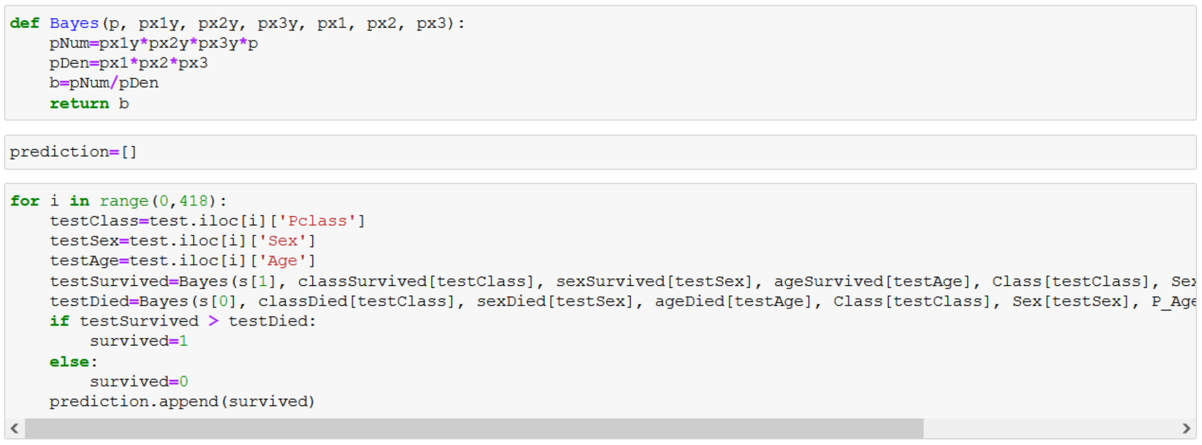
```
train.loc[train["Sex"] == "male", "Sex"] = 0
train.loc[train["Sex"] == "female", "Sex"] = 1

test.loc[test["Sex"] == "male", "Sex"] = 0
test.loc[test["Sex"] == "female", "Sex"] = 1
```

Figure 6: Numerical Values

After Data Cleaning, I implemented the process of naive bayes classification by finding the probability of features **Age**, **Sex**, **Pclass** based on its value counts.

6.3 Naive Bayes

After finding the probability of each feature in training data set i.e., age, sex and passenger class, I define a function for bayes theorem. Based on the probability between distinct features it will return a final value. So I made the test data set to pass through the defined function along with the probability which already done for training datasets.



```
def Bayes(p, px1y, px2y, px3y, px1, px2, px3):
    pNum=px1y*px2y*px3y*p
    pDen=px1*px2*px3
    b=pNum/pDen
    return b
```

```
prediction=[]
```

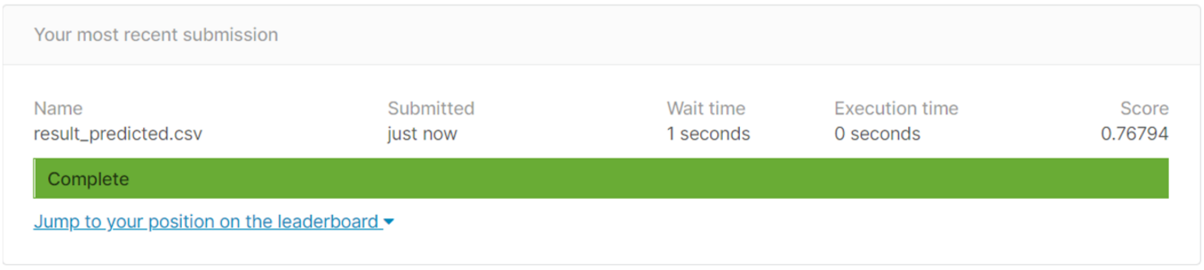
```
for i in range(0,418):
    testClass=test.iloc[i]['Pclass']
    testSex=test.iloc[i]['Sex']
    testAge=test.iloc[i]['Age']
    testSurvived=Bayes(s[1], classSurvived[testClass], sexSurvived[testSex], ageSurvived[testAge], Class[testClass], Se
    testDied=Bayes(s[0], classDied[testClass], sexDied[testSex], ageDied[testAge], Class[testClass], Sex[testSex], P_Ag
    if testSurvived > testDied:
        survived=1
    else:
        survived=0
    prediction.append(survived)
```

Figure 7: Bayes Implementation

With the **Bayes** function, I finally append all the returned values which result in the survived column for testing data.

Based on the value obtained, I created a excel file called **result_predicted.csv** with the columns namely, **Passenger id** and **Survived** of the test data set.

After submitting the result_predicted.csv in the Kaggle, I got the accuracy score of 0.76794.


Your most recent submission

Name	Submitted	Wait time	Execution time	Score
result_predicted.csv	just now	1 seconds	0 seconds	0.76794

Complete

Jump to your position on the leaderboard

Figure 8: Kaggle Accuracy

Also, I find accuracy by splitting the training dataset values from rows 100 to 300 for the **split** data frame. After that I created the data frame without **survived** column from the **split** data frame which is named as **final** data frame.

After passing the values of **final** data frame to the bayes function, we will get the survived values for the **final** data frame. With that survived values I find the true positive, true negative, false positive, false negative by comparing the **survived** values of **final** data frame with **split** data frame. Using the true positive, true negative, false positive and false negative, finally calculated the accuracy of the tested datasets. I got the accuracy score of 0.775, which then I converted to percentage so that I got 77.5% of accuracy. Also evaluated the values for Recall, FP-rate, Precision, F1-score.

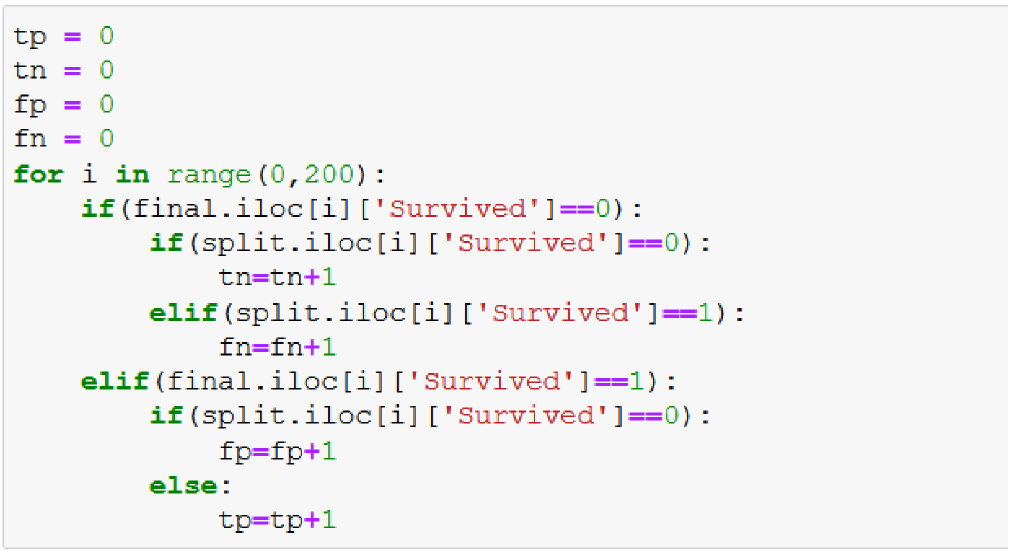

```
tp = 0
tn = 0
fp = 0
fn = 0
for i in range(0,200):
    if(final.iloc[i]['Survived']==0):
        if(split.iloc[i]['Survived']==0):
            tn=tn+1
        elif(split.iloc[i]['Survived']==1):
            fn=fn+1
    elif(final.iloc[i]['Survived']==1):
        if(split.iloc[i]['Survived']==0):
            fp=fp+1
        else:
            tp=tp+1
```

Figure 9: Performance Metrics

7 Further Improvements

Like all machine learning algorithms, we can boost the Naive Bayes classifier by applying some simple techniques to the dataset, like data pre-processing and feature selection. Here we have selected three features for survival prediction. They are Age, Passenger class, Sex. With these features we identified the survival prediction of test dataset. Also, to improve more accuracy we can select other features in the training dataset.

8 Conclusion

Here we have predicted the missing survived values of test dataset by training features in the training dataset with the predictive accuracy. We build a naïve bayes classifier for finding the survival of titanic disaster using bayes theorem by calculating the conditional probability.

9 Kaggle Link

Source Code : https://www.kaggle.com/manojprabakark/naive-bayes-classifier

10 Bibliography

Naive Bayes Classification

https://moodle.hof-university.de/pluginfile.php/271708/mod_resource/content/0/02_classification.pdf

https://moodle.hof-university.de/mod/resource/view.php?id=19874

Performance Metrics

https://moodle.hof-university.de/pluginfile.php/276605/mod_resource/content/0/04_ensemble.pdf

Datasets

https://www.kaggle.com/c/titanic

Anaconda Navigator for Python

https://www.anaconda.com/products/individual