



# **Technical Specification**

Name: Cian Manley

Student Number: 15563637

Supervisor: Mark Roantree

Project title: Classifier Analysis Application

# 1 TABLE OF CONTENTS

---

1	Table of Contents .....	2
1	Introduction .....	3
1.1	Abstract .....	3
1.2	Glossary .....	3
2	General Description .....	4
2.1	Motivation .....	4
2.2	User Assumptions .....	4
3	Research.....	5
3.1	Data Analysis.....	5
3.2	Naïve Bayes.....	5
3.3	Decision Tree .....	6
3.4	K-Nearest Neighbours .....	6
3.5	Support Vector Machines.....	6
3.6	Flask.....	7
4	Design .....	8
4.1	System Architecture .....	8
	Flask Application.....	9
	Filestore .....	9
	Graphical User Interface .....	9
4.2	High Level Design Diagram .....	10
5	Implementation.....	11
5.1	Python .....	11
5.2	Flask.....	13
5.3	HTML, Bootstrap & JavaScript .....	14
6	Problems Solved .....	15
6.1	Connecting Frontend to Backend .....	15
6.2	Storing CSV file.....	15
7	Results.....	16
7.1	Future Work.....	16
8	Reflections & Conclusion .....	17

# 1 INTRODUCTION

---

## 1.1 ABSTRACT

This project's aim was to create an easy to use application which provides users with a number of different options to get easy to read results from putting their data set through multiple different classification algorithms. This web application has an interactive graphical user interface by which provides user friendly interaction. It allows for easy optimisation and the running of algorithms. It produces results which are easy to read through a side-by-side comparison. The algorithms it uses throughout the application are Naïve Bayes, Decision Tree, K-Nearest Neighbours and Support Vector Machines. With these algorithms it provides the user with a handy toolkit to pass through their CSV files in order to compare how each of the algorithms performs on it. By allowing the user to return to the configurations page, the application lets the user rigorously perform analysis by using different metrics and being able to choose which attributes to use. This allows them to see how each attribute affects the target class.

## 1.2 GLOSSARY

<b>Predictive Analytics:</b>	This is branch of advanced analytics. It uses current data in order to make predictions about the future
<b>Flask:</b>	It is a micro web framework written in Python.
<b>K-Nearest Neighbours:</b>	A machine learning algorithm used for both classification and regression. It works based on minimum distance from the query to the training samples to determine the K-nearest neighbours.
<b>SVM:</b>	Support vector machines algorithm is a machine learning algorithm. A classification method that generates non-overlapping partitions and usually employs all attributes.
<b>Decision Trees:</b>	The decision trees algorithm builds models in the form of a tree structure. It works by continuously splitting the data into subsets in order to produce a sine curve with a set of rules.
<b>Naïve Bayes:</b>	Naïve Bayes is a classification algorithm based on Bayes' Theorem with an assumption of independence among predictors.
<b>GUI:</b>	Graphical user interface is a form of user interface which allows users to interact with different indicators and buttons.

## 2 GENERAL DESCRIPTION

---

### 2.1 MOTIVATION

Data Analytics is a huge field and has interesting developments which sort of caught my eye. With this curiosity, I became a bit interested in the field and start looking into it. Once I got my head around predictive analytics, which uses machine learning algorithms in order output a predictive result based on previous data, I gained a good interest in it. Firstly, one of my semester one projects was based on the prediction of a NFL game based on previous years data. It seemed like an easy task to accomplish as Naïve Bayes was a simple introduction to machine learning algorithms. Once I got Naïve Bayes working, I tried other classifiers in order to attempt to increase the accuracy.

When I went into work and had a chat with one of my colleagues about the project I was looking do, he gave me advice and how I would look about learning such a vast field. Talking to him influenced me to pursue this project in an effort to learn a different field than the one that I was no matter how daunting it may look.

I have also been looking at expanding my python skills and couldn't see a better opportunity then to attempt to make a program entirely based around it. Flask is such lightweight framework which allows for the easy implementation of external libraries. This sparked the thought that this will be a good use for this type of project.

### 2.2 USER ASSUMPTIONS

The target audience for this application is towards people looking to learn data mining and get involved in predictive analytics. In order to use the program, users will have to have some understanding of pre-processing in order to get the outcome they want from the application. There can be many issues that affect the outcome and it is up to the user to provide data that is in a suitable format to be put through a machine learning algorithm. When using the program, the only type of pre-processing which has been implemented is normalization. The user must know about data cleaning or data integration as it is not performed anywhere within the application.

To get full use of the application the user must know what kind of data they are putting through. This is because different types of data perform better using certain classifiers. Due to this they must know about the outputs and what they mean in order for them to gain any insight into what is so important about which algorithm. This means that the end user must have a bit of knowledge about Data Analytics but does not have to know how to program these models in themselves.

## 3 RESEARCH

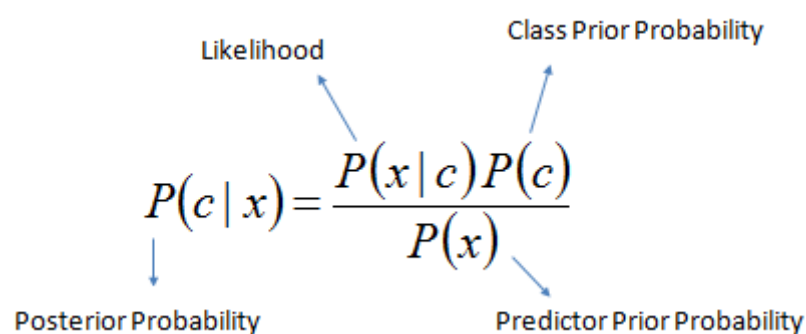
---

### 3.1 DATA ANALYSIS

When I first started this application, I had to begin researching somewhere to understand what I was developing. Data analytics is a large IT field which helps business make better informed decisions using previously acquired data. There are different types of analytics but the one I have been focusing on is Predictive Analytics. Predictive analytics uses different machine learning techniques, data and statistical algorithms in order to determine the likelihood of future results based on historical data. In the begin I had to research multiple different algorithms in order to decide which ones I needed to use. Once I chose four, a thorough knowledge of them was needed. Not only were there multiple different algorithms but there were also multiple implementations in order to improve accuracy within each one. In my attempt, I had to make a choice of a generalized version of each one to prevent misleading results.

### 3.2 NAÏVE BAYES

Naïve Bayes is a classification technique based on Bayes' Theorem. It has an assumption that there is independence among predictors. Naïve Bayes is seen as one of the easier algorithms to learn.



The diagram shows the Naïve Bayes formula with labels pointing to its components. The formula is 
$$P(c | x) = \frac{P(x | c)P(c)}{P(x)}$$
 The labels and their corresponding parts are: 'Likelihood' points to  $P(x | c)$ ; 'Class Prior Probability' points to  $P(c)$ ; 'Posterior Probability' points to  $P(c | x)$ ; and 'Predictor Prior Probability' points to  $P(x)$ .

At the start of the project, I had a basic grasp of Naïve Bayes. Implementing it was straight forward when using a set dataset. Naïve Bayes has a few different model types. The ones which were investigated where, Gaussian, Multinomial and Bernoulli. After looking into each of them, Gaussian was best suited to this application. Gaussian is the model used in classification, but it is assumed the features follow a normal distribution. This led to me then implementing normalization into the application in order to increase the overall outcome. It does this by adjusting the values contained within the dataset to a common scale.

### 3.3 DECISION TREE

Decision Tree is a classification technique using a tree-like graph of decisions and their possible consequences. Different techniques can be applied to Decision Tree in order to improve its accuracy rating. Two which I looked at were Information Gain and Gain Ratio. These metrics are used in considering how to split a node. The Gini index measurement is the probability of a random sample being classified incorrectly if we randomly pick a label according to the distribution in a branch. Whereas entropy is the measurement of information.

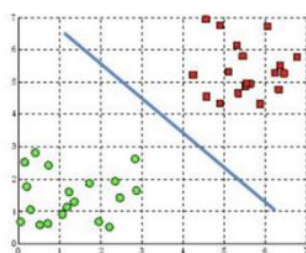
### 3.4 K-NEAREST NEIGHBOURS

K-Nearest Neighbours is a non-parametric, lazy learning algorithm. KNN is considered lazy due to it not using training data points to do any generalization. This lack of generalization means that KNN keeps all the training data. With KNN, it uses an input called the  $K$ -value. This  $K$  value is used to decide the number of neighbours used in the classification. There are also two different metrics that can be applied to the classification. There is the Euclidean Distance and Manhattan Distance which can be an applicable metric. It is used on deciding the distance between two different data points.

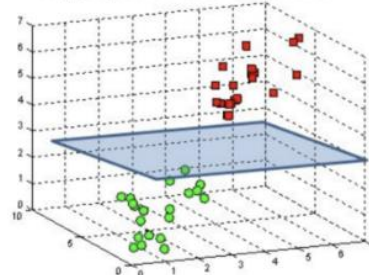
### 3.5 SUPPORT VECTOR MACHINES

Support Vector machines is a classification algorithm that is highly preferred by many experts as it produces a significant accuracy with less computation power. The main objective of SVM is to find the hyperplane in an  $N$ -dimensional space that distinctly classifies the data points. As seen below in the two examples, there are two classes in each case with a hyper plane in between them in order to split the classes as accurately as possible.

A hyperplane in  $\mathbb{R}^2$  is a line



A hyperplane in  $\mathbb{R}^3$  is a plane



This hyperplane can change as we are looking for the maximum margin between data points of two classes. There are many different ways in reducing this maximum margin, one of the main ones being the use of the loss function after regularization.

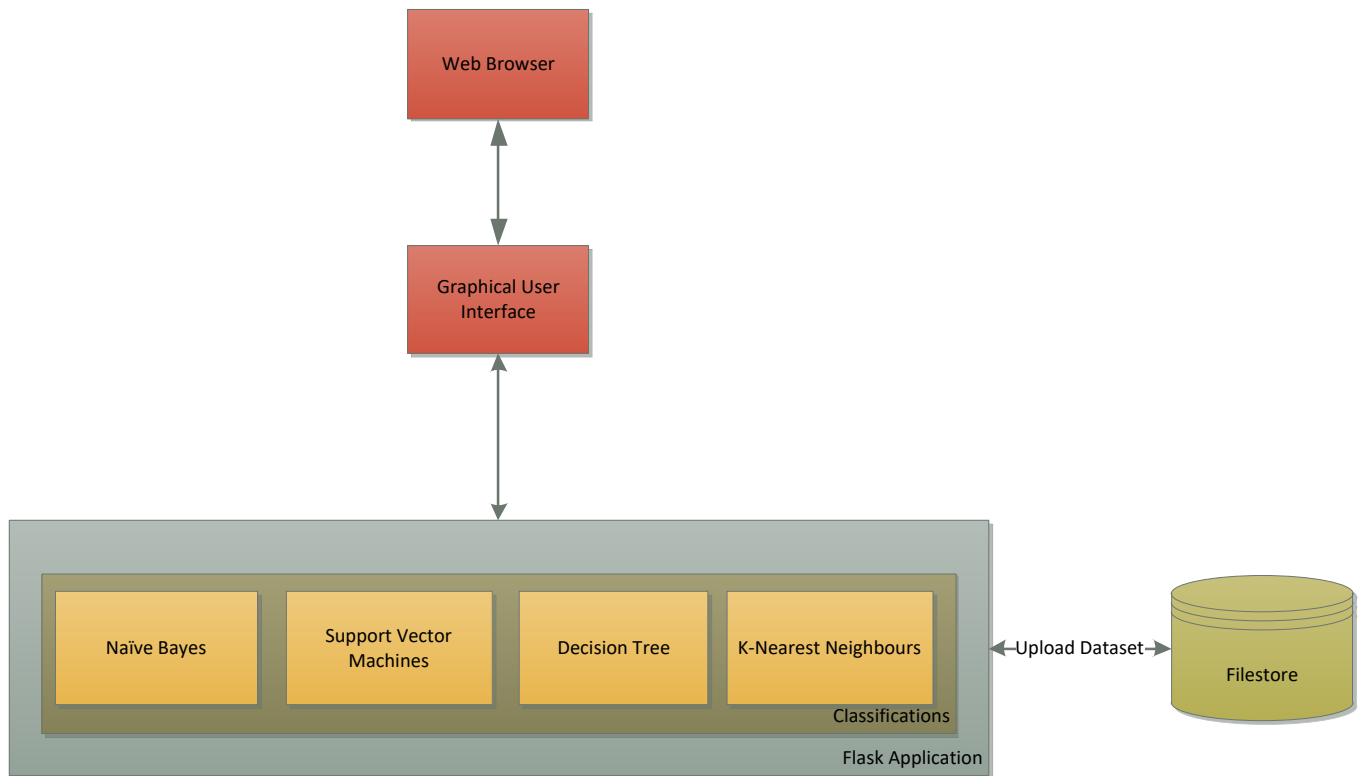
### 3.6 FLASK

I have never used a web framework quite like Flask before. This is my first implementation of this framework. The documentation that Flask has is excellent and provided me with a good base understanding of how to create and develop. From it being based on Python, I already had a decent understanding of it. This is what led me to believe that this was going to be a suitable framework to develop on as python has multiple different libraries to support machine learning and processing data. Not only does Python have these libraries but flask has a large number of extensions that integrate well into any project. Extensions such as *Flask-SQLAlchemy* and *Flask-WTF*. Both provide good management of parameters and data being passed through the web application.

## 4 DESIGN

---

### 4.1 SYSTEM ARCHITECTURE



The above shows the system architecture of the overall program. It is a simpler design than the one proposed in the functional specification as I had to remove React. This allowed for more development on the backend. In the backend it is made up of multiple modules. The classifications were contained within their own module, this was to allow for the easy implementation of more classification algorithms.



## Flask Application

The flask application has a number of different libraries used throughout in order to allow certain parts of the architecture to interact. Some of the libraries installed are, Flask-SQLAlchemy, Flask-WTF, Flask-Bootstrap and pygal. Each of these presented a purposed throughout the development of the application. Flask-WTF was used to store the different metrics in order for them to be used throughout the application. Pygal is used to display the graphic on the last page through the use of python instead of using a JavaScript library. Flask-Bootstrap is an interactive library used with Flask-WTF to apply decorators to each of the variables contained within the for.

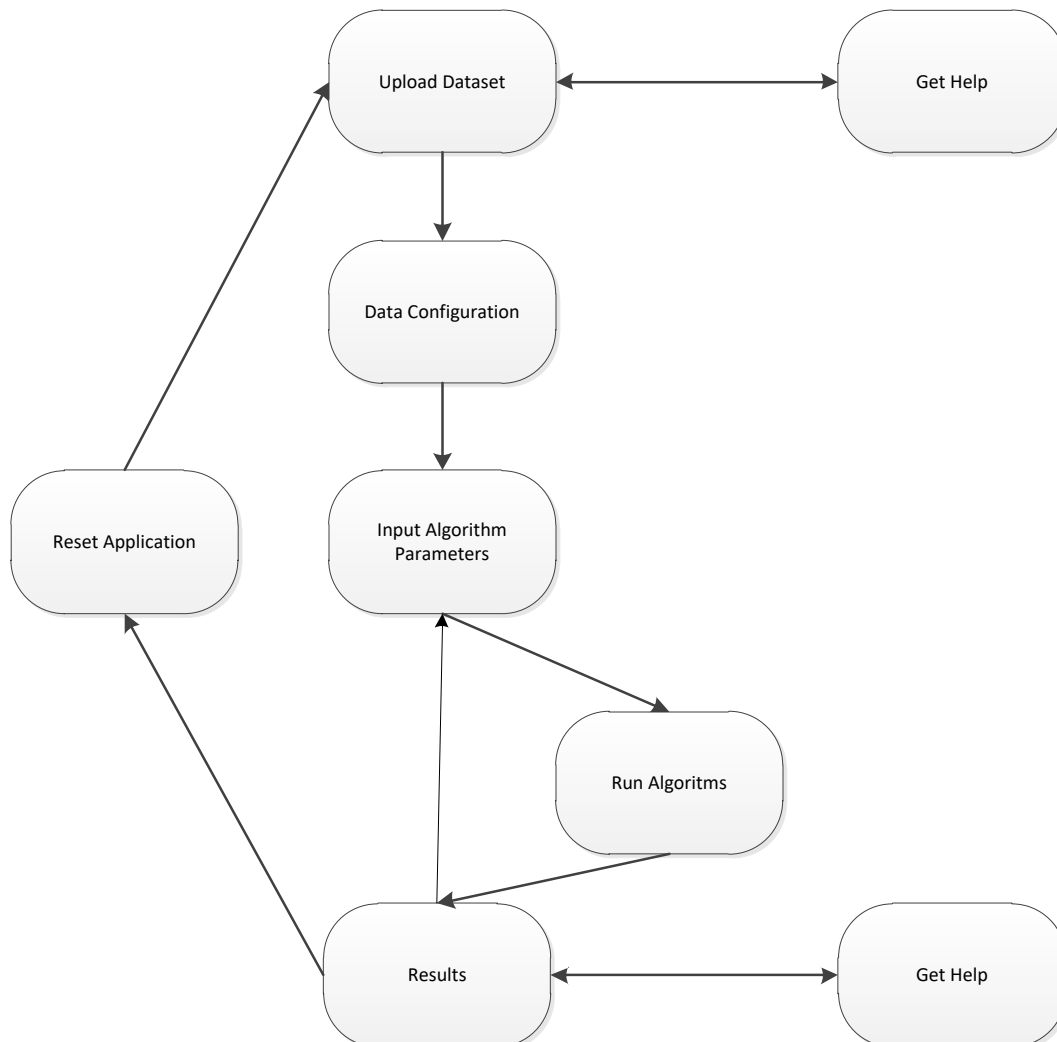
## Filestore

The Filestore was implemented using SQLite. This connects with the flask application using the extension Flask-SQLAlchemy. It allows for the use of querying the database and to be able to use the contents throughout the application. The database had to be created each time throughout development. If I was to scale up to production I would have to more than likely use a server. If I was to pursue this, postgresql would be the optimum choice.

## Graphical User Interface

The Graphical User Interface is the at the frontend of the project. It allows for easy use of the application. It is only a small component but is written using HTML5, CSS, JavaScript and Bootstrap. There are only tiny script tags used throughout the application in order to provide the user with a more friendly experience.

## 4.2 HIGH LEVEL DESIGN DIAGRAM



The above diagram illustrates the high-level design of the web application. When first entering the application, a route containing a file upload and set of different parameters on how the dataset is processed. This is where the user uploads the dataset and an enters in the parameters, if successfully submitted they are directed to the form page by which they choose which algorithms they want to use and their corresponding metrics. The about page is available throughout the application and the user is able to return to where they were without having to upload once again and start from the beginning. Once all, the metrics are entered, its is put through the selected modules and brought to the results page to which the user can see their expected output. When they

are finished there is a button which resets the application and starts the user from the beginning, or they can go back to the form page in order to enter different metrics.

## 5 IMPLEMENTATION

---

### 5.1 PYTHON

```
def form():
    form = MetricForm()
    Cform = ConfigForm()

    file = request.files['inputFile']

    newFile = Filestorage(name=file.filename, data=file.read())
    db.session.add(newFile)
    db.session.commit()

    file_data = Filestorage.query.filter_by(id=1).first()
    csv_file = pd.read_csv(BytesIO(file_data.data))

    if Cform.header_check.data == 'No':
        csv_file = addHeaders(csv_file)
        headers = list(csv_file)
    else:
        headers = list(csv_file)

    header_l = len(headers)

    choices = [(headers[i], headers[i]) for i in range(len(headers))]

    form.selectedAtt.choices = choices
    form.targetAtt.choices = choices

    if request.method == 'POST' and form.validate_on_submit():
        return render_template('result.html')
    else:
        return render_template('form.html', form=form, headers=headers)
```

The Python language was the part I was most familiar with when starting this project. I choose this as the main language due expansive libraries relating to machine learning. Each of the models are taken from the SKLearn libraries. Each has been separated into its own python file. This is done in order to keep track and add any additional metrics if needed. Most of the issue came from the processing of the data. In order to allow for any type of data to be used I had to implement an addHeaders function. This is done to give datasets without column names, names. It was simply achieved as the conversion from csv to data frame using the pandas library. Once converted to a data frame the csv files are easily managed. It allows for simple selection of the target class and columns which are to be used.

```

def stripColumns(dataset, chosenCols, predictCol):
    X = dataset[chosenCols]
    y = dataset[predictCol]

    return X, y

def splitData(X, y, splitData):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=splitData, random_state=1)

    return X_train, X_test, y_train, y_test

def kNeigh(csv, features, target, k, metric, split):
    X, y = stripColumns(csv, features, target)
    X_train, X_test, y_train, y_test = splitData(X, y, split)

    #Add in attribute selection, splitter and/or max_depth in order to optimize

    if metric == 'manhattan':
        #P=1 in order to use Manhattan distance
        classifier = KNeighborsClassifier(n_neighbors=k, p=1)
    else:
        #P=2 in order to use Euclidean distance
        classifier = KNeighborsClassifier(n_neighbors=k, p=2)

    classifier.fit(X_train, y_train)

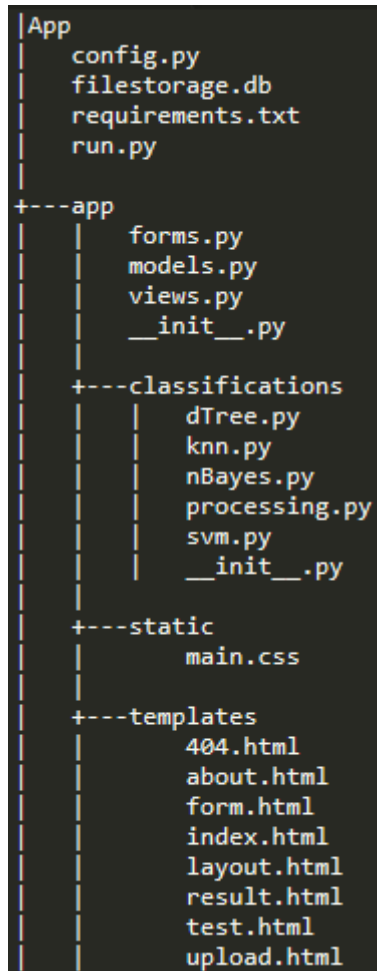
    y_pred = classifier.predict(X_test)

    return metrics.accuracy_score(y_test, y_pred)

```

This is the one of the implementations of a classification algorithm. The K-nearest neighbours function take in the different feature classes and target class. Once all this is inputted it strip the columns to remove ones that are not needed, these are then put through the splitData function in order to get its different test and training data. Once this data is passed through it fits the data to the model. It then performs classification using predict on the model using the test data. Lastly it outputs the accuracy score from the data being passed through the model.

## 5.2 FLASK



Flask was a new lightweight framework for me to learn. It is a microframework which doesn't require particular tools and libraries to operate. It is a useful framework as there are multiple third-party form validation and database integration libraries that are suited to your project.

This is a tree diagram of the applications file structure. It separates each of the components into different sections and modules. The config.py file contains the variables which the application may need to run. Filestorage.db is the static database setup by SQLite in order to store uploaded files for use throughout the program. Run.py is the file that is told to run when starting up the server. The in the app it contains multiple different files that initials forms and databases.

Below is an example of the use of Flask-WTF. It allows the program the use of different variables throughout the application. This allows for easy navigation of the web application and provides a user-friendly experience. The main benefit of this is to allow, users to return to the form page in order to change the different inputs to allow the user to see how the data performs on different metrics or different chosen columns.

```
from flask_wtf.file import FileField, FileRequired, FileAllowed
from wtforms.widgets import ListWidget, CheckboxInput

class MultiCheckboxField(SelectMultipleField):
    widget = ListWidget(prefix_label=False)
    option_widget = CheckboxInput()

class ConfigForm(FlaskForm):
    file = FileField(validators=[FileRequired(), FileAllowed(['csv'], 'CSV Only!')])
    header_check = RadioField('headers', choices=[('Yes', 'Yes'), ('No', 'No')])
    split = IntegerField('Train Test Split', validators=[InputRequired()])

class MetricForm(FlaskForm):
    selectedAtt = SelectMultipleField('selectAtt', choices=[])
    targetAtt = SelectField('targetAtt', choices=[])
    nbCheck = BooleanField('Naive Bayes', default=False)
    dtCheck = BooleanField('Decision Tree', default=False)
    knnCheck = BooleanField('K-Nearest Neighbours', default=False)
    svmCheck = BooleanField('Support Vector Machine', default=False)
    knnMetric = RadioField('Distance Method', choices=[('manhattan', 'Manhattan'), ('euclidean', 'Euclidean')], default='euclidean')
    kValue = IntegerField('K-Value', validators=[InputRequired()])
    dtMetric = RadioField('Distance Method', choices=[('entropy', 'Entropy'), ('gini', 'Gini Index')], default='entropy')
```

## 5.3 HTML, BOOTSTRAP & JAVASCRIPT

For my frontend was developed using HTML5 and Bootstrap. The main theme that was used was Bootstrap 4. This allowed for smooth development without the need for CSS files. Once the GUI was setup, I made a CSS in order to tweak some little features to suit the application. The file 'layout.html' contains a few different designs. I set this up in order to have a static navbar throughout the program. Also, throughout the application I made use of some small JavaScript script tags. This was to allow little features such as the below. This script changed the label "custom-file-label" to the filename once it was selected in the browse field. Below is a screenshot of code used throughout the frontend.

```
{% extends 'layout.html' %}
{% import 'bootstrap/wtf.html' as wtf%}
{% block content %}

<h1>Upload an CSV file</h1>
<hr>

<div class="container h-100">
  <div class="row h-100 justify-content-center align-items-center">
    <div class="col-12">
      <form action="/form" method="POST" enctype="multipart/form-data">
        <div class="form-group">
          <div class="custom-file">
            <input type="file" name="inputFile" class='custom-file-input'>
            <label class="custom-file-label" for="inputFile">Select CSV...</label>
          </div>
        </div>
        <div class="form-row">
          <div class="form-group col-md-6">
            <label>Does your dataset have Headings?</label>
            {{ wtf.form_field(form.header_check) }}
          </div>
          <div class="form-group col-md-6">
            {{ wtf.form_field( form.split ) }}
          </div>
        </div>
        <button type="submit" class="btn btn-primary">Upload </button>
      </form>
    </div>
  </div>
</div>

<script type="application/javascript">|
  $('input[type="file"]').change(function(e){
    var fileName = e.target.files[0].name;
    $('.custom-file-label').html(fileName);
  });
</script>

{% endblock content %}
```

## 6 PROBLEMS SOLVED

---

It felt like this project had issues day in day out as I was learning the flask web framework along side the development of the project. Some of the errors were only minor which maybe included setting up forms as I wasn't probably too sure how to use them but quickly figured it out.

### 6.1 CONNECTING FRONTEND TO BACKEND

When attempting to follow my initial design by using a React frontend to create a much more minimalistic and simpler GUI, I ran into multiple issues. Having a decent knowledge of Python allowed me to get the backend running smoothly but connecting it to the frontend became a bit too difficult as certain parameters were not being received and files were not being uploaded to the file store. I attempted to solve these issues throughout April, but deadline was coming, and I had to make a decision to change the structure of my System Architecture.

After consideration I decided to make it a large Flask based application. The change didn't take too long as I had a basic boilerplate set up in HTML5 in order to make the switch. I now had a basic frontend using HTML5, CSS and Bootstrap.

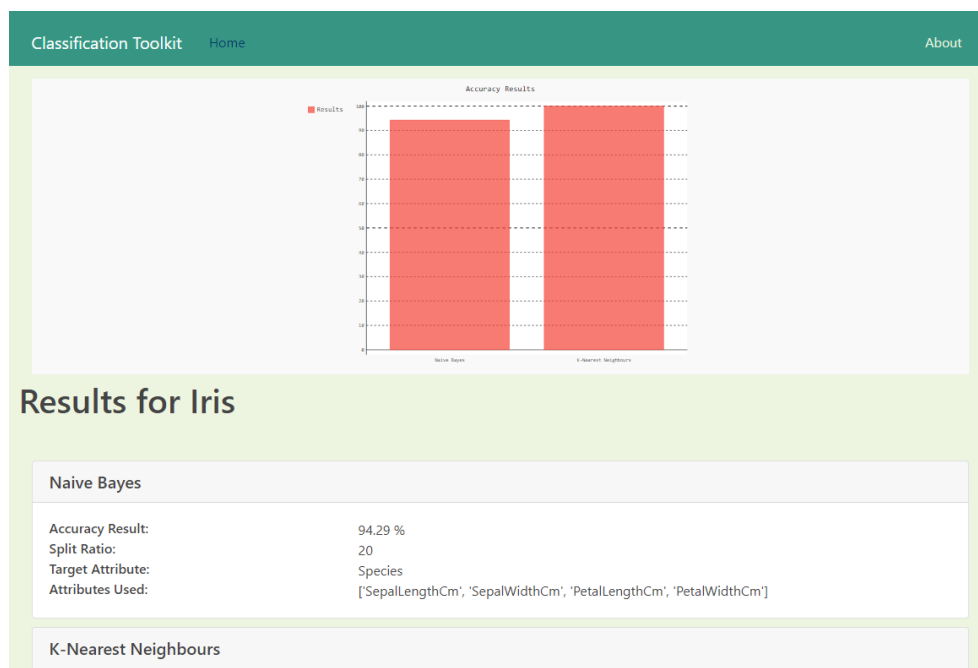
### 6.2 STORING CSV FILE

Storing the CSV file became an issue from the very beginning. While implementing React, I sent the file to my backend by converting it to a JSON file in order to be read in. Once the file got to the backend, it was properly reading the data in and I couldn't process the data. This was a big obstacle in my project as I needed to use these files throughout the application. This was one of the other factors which led me to change my design.

Once I moved onto the large Flask application structure, it became significantly easier to manage these files. At first, I attempted use the files without storing them at first through the use of the package Flask-WTF. This is package which aids in the processing of forms throughout a Flask application. It worked for what I had at the time, but I needed to keep the file stored throughout the session. Flask-SQLAlchemy was the next package which provided smooth integration of sqlite database into the project. Once this was used, I was able to use this file throughout the application.

## 7 RESULTS

---



These results are produced are easy to read and are displayed in a side by side manor to allow for comparisons between the multiple algorithms used.

### 7.1 FUTURE WORK

There is a lot of room with this project as Data Analytics is a large field with multiple different classifiers and techniques to improve the predictive analytics of different types of Data. This project contains the four different classifiers, but more can be added on.

### Multiple Files

Once I implemented the SQLite database into the application. It became viable to upload multiple files at once in order select what to do with each of them. Due to time limitations and only getting the level of knowledge needed I was unable to implement this functionality in this version. To go about this, I would give each of the files and unique number and store them in the database as a Large Binary file. The upload page would need a separate page in order to load the different files onto the main page. Then lastly, each of the uploaded files would have to have an icon in order to click to use. Maybe this functionality can go even further and run on multiple different files at once.



## New Classifiers

There is any number of new classification algorithms. Even the ones used can have new metrics used to increase the accuracy. While researching Support Vector Machines, I investigated the use of kernels. Kernels are used to transform the input data into a required form. It applies different mathematical functions in order to make certain data more suitable to processing. Some other classifications that could be implemented are Random Forest, Boosted Trees and Linear Regression. Each of these can be programmed in easily as each of the classifiers are contained within a module, and then a few added fields into the GUI and everything should be good to go.

## Program my own Models

At the beginning I attempted to program my own modules as I was learning them using Jupyter Notebooks. It was simple enough to create my own models as I had a static csv file with the same inputs. Once I began using the Naïve Bayes model in the original desktop application it took some time to process the different csv files. After this, I began to think, why re-invent the wheel and began to use SKLearn to apply their classifier models and speed up the process. Due to this, I felt like I skipped out on learning certain parts of algorithms that could have benefitted me in the future.

## 8 REFLECTIONS & CONCLUSION

---

Overall, I'm pretty underwhelmed by my overall production of this application. I took me a bit longer than anticipated to start. Once, I delved more deeply in the data mining side, I became curious and this curiosity motivated me throughout the rest of the project. I achieved what I set out to do and not much more. This why towards the end, I started to notice more improvements that could've been made. However, I feel like I have accomplished at making a user-friendly application for which people can use.

In the learning side, Flask has become something quite known to me now and I enjoy that there is so much to it. I have definitely gained the knowledge of different parts of Data Analysis especially Predictive Analytics. Data Analytics has become a field that I have become quite interested in now and may just pursue in my future endeavours.