**A PROJECT REPORT**

**on**

**“Predicting Spine Abnormalities using Machine Learning”**

**Submitted to**

**KIIT Deemed to be University**

**In Partial Fulfillment of the Requirement for the Award of**

**BACHELOR’S DEGREE IN**

**INFORMATION TECHNOLOGY**

**BY**

**Kush Jayank Pandya**

**Harshit Agarwal**

**Daibik DasGupta**

**Biswajeet Sahoo**

1705701

1705697

1705692

1705689

**UNDER THE GUIDANCE OF**

**Manjusha Pandey**

****

**SCHOOL OF COMPUTER ENGINEERING**

**KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY**

**BHUBANESWAR, ODISHA - 751024**

**May 2020**

A PROJECT REPORT

on

“Predicting Spine Abnormalities using Machine Learning”

Submitted to

KIIT Deemed to be University

In Partial Fulfillment of the Requirement for the Award of

BACHELOR’S DEGREE IN INFORMATION TECHNOLOGY

BY

Kush Jayank Pandya

Harshit Agarwal

Daibik DasGupta

Biswajeet Sahoo

1705701

1705697

1705692

1705689

UNDER THE GUIDANCE OF

Manjusha Pandey



SCHOOL OF COMPUTER ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

BHUBANESWAE, ODISHA -751024

May 2020

KIIT Deemed to be University

School of Computer Engineering

Bhubaneswar, ODISHA 751024



CERTIFICATE

This is certify that the project entitled

“Predicting Spine Abnormalities using Machine Learning“

submitted by

Kush Jayank Pandya Harshit Agarwal

Daibik DasGupta

Biswajeet Sahoo

1705701

1705697

1705692

1705689

is a record of bonafide work carried out by them, in the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science & Engineering OR Information Technology) at KIIT Deemed to be university, Bhubaneswar. This work is done during year 2019-2020, under our guidance.

Date: / /

Manjusha Pandey

Project Guide

**Acknowledgement**

It is with a sense of satisfaction that our group is able to complete and compile this project. However, it would not have been possible without the aid from our project guide. We would like to express our gratitude to **Professor Manjusha Pandey**, whose advice and guidance has proven invaluable in bringing about the end product. It is through her constant encouragement that we have been able to see this project to its completion.

Kush Jayank Pandya

Harshit Agarwal

Daibik DasGupta

Biswajeet Sahoo

**ABSTRACT**

Lower back pain (LBP) is caused because of assorted reasons involving body parts such as the interconnected network of spinal cord, nerves, bones, discs or tendons in the lumbar spine. LBP is pain, muscle pressure or stiffness localized underneath the costal edge or more the substandard gluteal folds, with or without leg torment for the most part sciatica, and is characterized as endless when it holds on for 12 weeks or more then again, non-particular LBP is tormented not credited to an unmistakable pathology such as infection, tumour, osteoporosis, rheumatoid arthritis, fracture, or inflammation.[1] Hence aiming to look for preventive measure rather than curative, this study suggests a classification methodology for Chronic LBP disorder using Machine Learning techniques.

We are going to have an exploration of this data and then predict lower back pain using these biometric measurements; this is a binary classification problem, as we have two outcomes: abnormal and normal. There are many ways that we can approach a classification problem, and there are many algorithms that can be used to create predictions. In this report, we use logistic regression, random forests and TensorFlow Lib. We compare the accuracies and provide a method for looking at other predictive metrics and what they mean when evaluating a model by building the equivalent GUI.

**Keywords:** Spine Abnormalities, Chronic LBP disorder, LBP Prediction,

Machine Learning, TensorFlow, Logistic Regression, Random Forests.

Contents

Chapter 1

Introduction

* 1. Problem Statement

Lower Back Pain is one of the most commonly occurring ailments in the human population, due to a variety of reasons from infection, tumour, osteoporosis, rheumatoid arthritis, fracture, or inflammation. These issues are very much commonplace and as such Lower Back Pain affects a large range of people. The biological cause for it can come from a variety of different sources too due to the interconnected network of spinal cord, nerves, bones, discs or tendons in the lumbar spine.[1] Problems with the spine are very difficult to cure without lengthy and strenuous treatment and expenditure of a large amount of time and money. This is why a preventative measure is far superior than a curative measure.

* 1. Project Description

We are going to have an exploration of this data and then predict lower back pain using these biometric measurements; this is a binary classification problem, as we have two outcomes: abnormal and normal. There are many ways that we can approach a classification problem, and there are many algorithms that can be used to create predictions. In this report, we use logistic regression, random forests and TensorFlow Lib. We compare the accuracies and provide a method for looking at other predictive metrics and what they mean when evaluating a model by building the equivalent GUI.

This project is being made as a minor project by KIIT students. This is a Machine Learning Project. We have collected the Dataset, preprocessed it. And on that data, We found out the best model for it and Deployed it.

* 1. Future Scope

If any abnormality in a person’s spine is detected at an early stage then the patient can be treated to completely eliminate the problem before it leads to further complications. With the help of Machine Learning, we can create technology that can detect these abnormalities at an early stage.

Chapter 2

Literature Survey

2.1 Research Material

The following medical research and machine learning repositories were studied to form a basis of the solution for our problem statement.

* 1 P. Brinckmann, “Pathology of the vertebral column,” Ergonomics, vol. 28, no. 1, pp. 77–80, 1985.
* Z. Ahmad, R. Mobasheri, T. Das, S. Vaidya, S. Mallik, M. ElHussainy, and A. Casey, “How to interpret computed tomography of the lumbar spine,” Annals of the Royal College of Surgeons of England, vol. 96, no. 7. pp. 502–507, 2014.
* A. Liguori, F. Galli, M. Gurgitano, A. Borelli, M. Pandolfi, F. Caranci, A. M. Magenta Biasina, G. G. M. Pompili, C. L. Piccolo, V. Miele, C. Masciocchi, and G. Carrafiello, “Clinical and instrumental assessment of herniated discs after nucleoplasty: A preliminary study,” Acta Biomed., vol. 89, pp. 220–229, 2018.
* Kaggle Inc., “The Home of Data Science,” Kaggle is the world’s largest community of data scientists, 2014. [Online]. Available: http://www.kaggle.com.
* K. Bache and M. Lichman, “UCI Machine Learning Repository,” University of California Irvine School of Information, vol. 2008, no. 14/8. p. 0, 2013.
* L. Breiman, “Random forests,” Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001.

Chapter 3

Software Requirements Specification

3.1 Overall Description

3.1.1 Product Perspective

# The tool used for performing analysis on the data is Python. The aim was to create a classification model with higher accuracy than existing research done. Out of 310 observations, 67% of the data was used for training and 33% of the data was used for testing. The sample function was run on the entire dataset to generate sets of training and testing data. This was done to get a performance score of each model generated using all of the classification methods used in this research.

3.1.2 Product Perspective

This product shows us whether the spine details you added are normal or abnormal. The user is needed to input the value such as pelvic\_incidence, pelvic tilt, etc. into the Application. And submit it and the application gives out results.

3.1.3 Technology Used

Programming Language - Python

Ide - Jupyter Notebook, Google collab

Third-party libraries - pandas, NumPy, matplotlib, TensorFlow,pyinstaller, Tkinter

Technology type - Machine Learning, Deep Learning, GUI

3.1.4 Operating Environment

This application can run on any Desktop PC. There is no internet connection required for its main task. But for the help button, it will be redirected to a webpage.

3.1.5 Design and Implementation Constraints

The most important Implementation Constraint is the Dataset on which we would be training our ML models. If the models are not up to the required accuracy. It will be a big problem. Also, the dataset should be mostly complete. Some data missing in an example might lead us to not consider it while training, leading to a smaller dataset.

3.1.6 User Documentation

A small help page showing how our software works and what it does will be put up on GITHUB. It will describe the project Medical Importance and its working.

3.1.7 Assumptions and Dependencies

The whole project is done on python by using many third-party libraries. So Design and implementation are very much dependent on them. For example, TensorFlow, a Deep Learning Library can perform very well with GPU support. And its Installation is a hard task to implement.

Also, a Third-party library called pyinstaller which converts the python code into an executable program is also an important dependency without this working properly we will not be able to convert our python code into an executable file.

3.2 External Interface Requirements

3.2.1 User Interface

The User Interface is kept as simple as possible. There would be two buttons (i.e Help and Submit). Along with this, there would be twelve entry boxes where users have to input different attributes labeled around them. ‘Submit’ button will open a different window displaying the result. And with the ‘help’ button you would be directed to a webpage. The whole of the UI will be made using a Tkinter.

3.2.2 Hardware Interfaces

The hardware can run on a PC environment supported by Pyinstaller.

Chapter 4

Requirement Analysis

4.1 Functional Requirements

4.1.1 Highly Accurate ML Model

The Application makes sense only if we are correctly able to predict the outcome. It’s a High priority feature with the relative risk being 5.

To find this model we would have to iterate over many approaches using a caviar approach to find the best model fit. We use Scikit-learn and TensorFlow libraries to do this. It also should be feed data which is clean and consistent which is done by the Pandas library.

4.1.2 Decent GUI

A simple and fast GUI. It’s a High priority feature with the relative risk being 1.

To achieve this we would use Tkinter. And does not have any more functional Requirement.

4.2 Non-Functional Requirements

4.2.1 Help Page

A simple help webpage will be uploaded on Github which can be utilized by the user in case of complications.

4.3 Future Enhancements

Since we are dealing with a Medical Problem, it will be more suitable for the end-user to have easier access to this. So, we can deploy the model on an Android App and with the help of an API.

Chapter 5

System Design

5.1 Basic Framework

The basic working principle of the project is dependant upon Machine Learning algorithms through which we analyze the biometric details of a patient’s spinal system and run these values through an optimized algorithm that we have found to give the most accurate results and then identifying whether the patient may suffer from an abnormal spinal health condition or not.

The implementation of the Machine Learning algorithm is done through Python mainly and by making use of third-party libraries that are commonly used for Machine Learning projects to make working much more easier. They are:

* **NumPy:** It is a library for the Python programming language, adding support for large, multidimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.[2]
* **Pandas:** It is an open source data analysis and manipulation tool, built on top of the Python programming language.[3]
* **Scikit-learn:** Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN.[4]
* **Matplotlib:** Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.[7]
* **TensorFlow:** It is a software library for data flow and differentiable programming across a range of tasks.[5]
* **Plotly:** Used for production of the front-end for ML and data science models.

5.2 Machine Learning Model

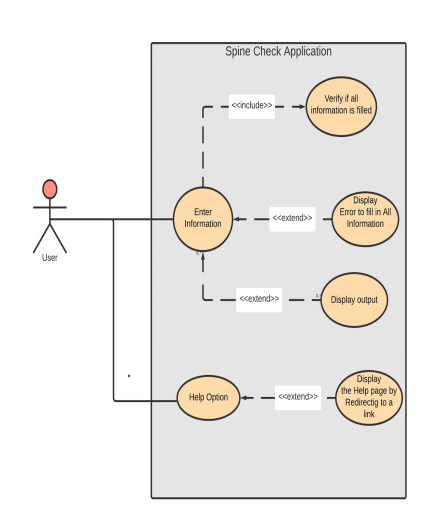
After testing by various models, recording the predicted output and comparing it for validation, we have found the best suited Machine Learning Model. Finally, we also further improve our prediction models by testing in various parameters for any given selected model until the best prediction accuracy is found. As the problem is requires a binary input, we have utilized classification algorithms to predict any entity into either Normal or Abnormal classification.

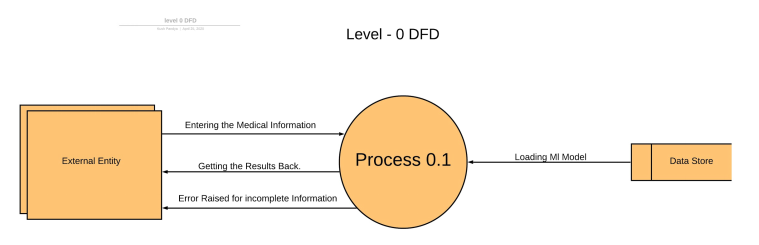
5.3 Stand-Alone Application

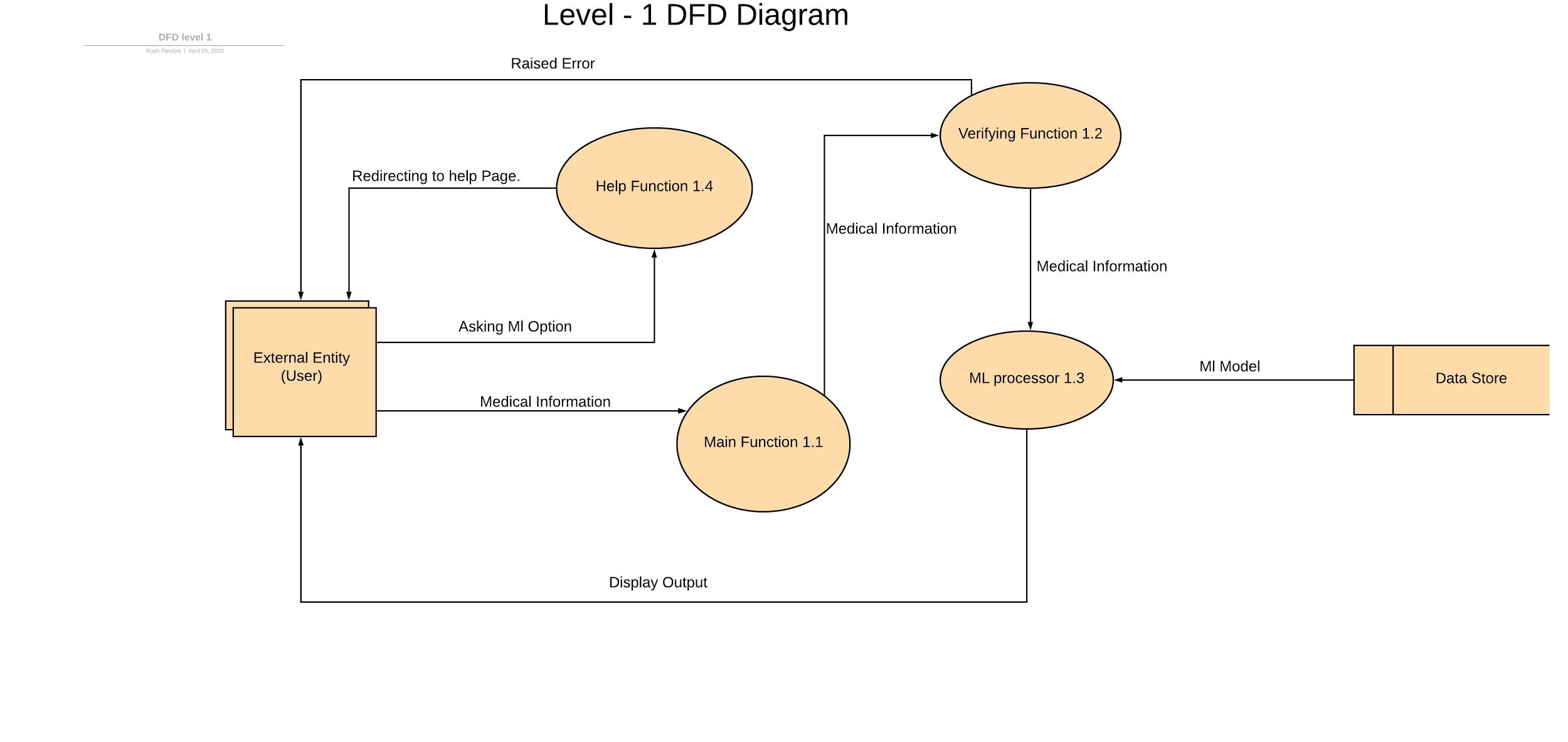
The following library was used to produce a stand-alone executable file that the user can run with ease and utilize the features of our project.

* **Pyinstaller:** freezes (packages) Python applications into stand-alone executables, under Windows, GNU/Linux, Mac OS X, FreeBSD, Solaris and AIX.

5.4 UML Diagrams







Chapter 6

System Testing

The system testing is performed on the same data set consisting of biometric data of the spine, but with varying models to see which model provides the best result in regards to accuracy of prediction.

6.1 Decision Tree Classifier

The following testing was done using the Decision Tree Classifier Model. ‘depth’ parameter is used and deeper the tree, the more splits it has and it captures more information about the data.

|  |  |  |  |
| --- | --- | --- | --- |
| Test ID | Test Conditions | Train Accuracy | Test Accuracy |
| T0101 | Depth 1 | 0.782 | 0.669 |
| T0201 | Depth 2 | 0.884 | 0.766 |
| T0301 | Depth 3 | 0.937 | 0.815 |
| T0401 | Depth 4 | 0.971 | 0.766 |
| T0501 | Depth 5 | 1.0 | 0.796 |
| T0601 | Depth 6 | 1.0 | 0.815 |
| T0701 | Depth 7 | 1.0 | 0.834 |
| T0801 | Depth 8 | 1.0 | 0.815 |
| T0901 | Depth 9 | 1.0 | 0.815 |

6.2 Random Forest Classifier

The following testing was done using the Random Forest Classifier Model. This model uses ‘depth’ as its parameter which represents the depth of each tree in the forest. The deeper the tree, the more splits it has and it captures more information about the data.

|  |  |  |  |
| --- | --- | --- | --- |
| Test ID | Test Conditions | Train Accuracy | Test Accuracy |
| T0102 | Depth 1 | 0.782 | 0.669 |
| T0202 | Depth 2 | 0.884 | 0.766 |
| T0302 | Depth 3 | 0.937 | 0.815 |
| T0402 | Depth 4 | 0.971 | 0.766 |
| T0502 | Depth 5 | 1.0 | 0.796 |
| T0602 | Depth 6 | 1.0 | 0.815 |
| T0702 | Depth 7 | 1.0 | 0.834 |
| T0802 | Depth 8 | 1.0 | 0.815 |
| T0902 | Depth 9 | 1.0 | 0.815 |
| T1002 | Depth 13 | 1.0 | 0.834 |

6.3 Gradient Boosting Classifier

The following testing was done using the Gradient Boosting Classifier.

|  |  |  |
| --- | --- | --- |
| Test ID | Train Accuracy | Test Accuracy |
| T0103 | 1.0 | 0.786 |

6.4 MLP Classifier

The following testing was done using the MLP Classifier.

|  |  |  |
| --- | --- | --- |
| Test ID | Train Accuracy | Test Accuracy |
| T0104 | 0.975 | 0.737 |

6.5 Support Vector Classifier

The following testing was done using the Support Vector Classifier.

|  |  |  |
| --- | --- | --- |
| Test ID | Train Accuracy | Test Accuracy |
| T0105 | 0.681 | 0.669 |

6.6 K-Neighbors Classifier

The following testing was done using the K-Neighbors Classifier.

|  |  |  |  |
| --- | --- | --- | --- |
| Test ID | Test Conditions | Train Accuracy | Test Accuracy |
| T0706 | Neighbor 7 | 0.855 | 0.611 |
| T1206 | Neighbor 12 | 0.821 | 0.640 |
| T1306 | Neighbor 13 | 0.840 | 0.621 |
| T1406 | Neighbor 14 | 0.797 | 0.650 |
| T1506 | Neighbor 15 | 0.811 | 0.621 |

6.7 TensorFlow Sequential Classifier

The following testing was done using the TensorFlow Sequential Classifier.

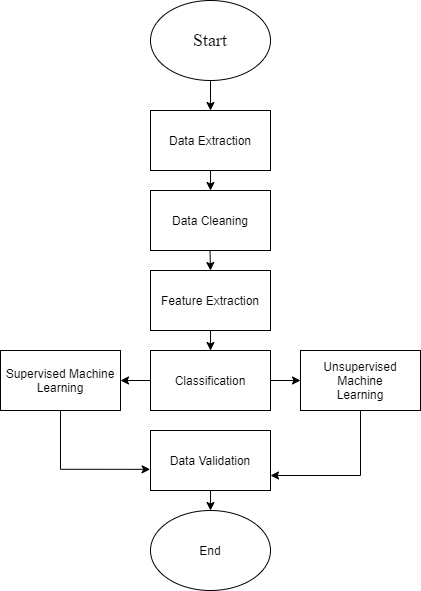
|  |  |  |
| --- | --- | --- |
| Test ID | Test Conditions | Test Accuracy |
| T0107 | 310/1 | 0.964 |

Chapter 7

Project Planning

7.1 Methodology for Machine Learning Model

A simple flowchart of the standard methodology is given below. The provided flow chart forms the basis of most Machine Learning operations though depending on the conditions, a step may be minimized or expanded. For example, for a dataset that has already been cleaned and provided to the researcher, the data cleaning and transformation part of data extraction is thoroughly minimized. In our current project, we follow each and every step in order to produce the output required by the problem statement.



7.1.1 Data Extraction

Data extraction is a process that involves retrieval of data from various sources. Frequently, companies extract data in order to process it further, migrate the data to a data repository (such as a data warehouse or a data lake) or to further analyze it. It’s common to transform the data as a part of this process. For example, you might want to perform calculations on the data — such as aggregating sales data — and store those results in the data warehouse.[6]

In the case of our project, in the data set, all the attributes are in the numerical attribute. The dataset contains the classification of the patients classified into one of two categories: Normal (100 patients) or Abnormal (210 patients). Every patient is represented as a pattern with 12 bio-mechanical attributes, according to the following physical parameters: pelvic incidence, pelvic tilt, lumbar lordosis angle, sacral slope, pelvic radius, spondylolisthesis, pelvic slope, direct tilt, thoracic slope, cervical tilt, sacrum angle, scoliosis slope.

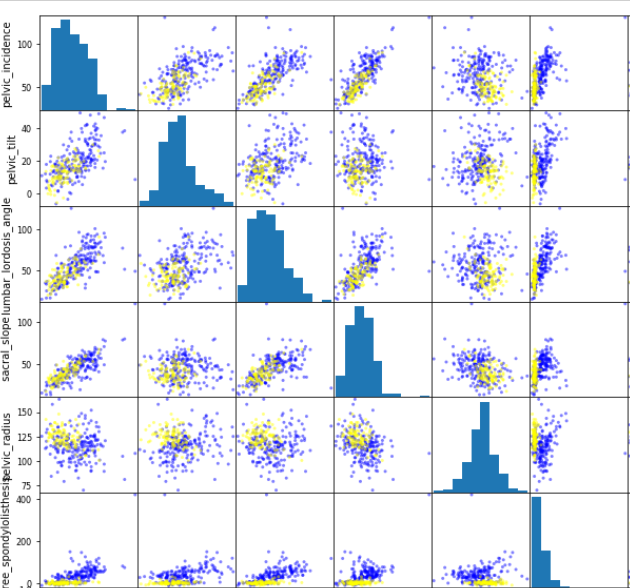
7.1.2 Data Cleaning

Data cleansing or data cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data.[6] In case of our project, we alter the attribute headers to their corresponding specifications within the CSV file. Following this, we observe the relevance of every attribute and perform data explorations within the values to find out any kind of valuable insight and accordingly decide the next course of action.

7.1.3 Feature Extraction

Feature extraction is a process of dimensionality reduction by which an initial set of raw data is reduced to more manageable groups for processing. A characteristic of these large data sets is a large number of variables that require a lot of computing resources to process.[6]

In our case, we have finally split the data set into the necessary testing and training sets after randomizing the order of the data set to prevent bias, and finally we have utilized Min Max Scaler to perform feature scaling on the entire dataset to reduce the range of values to a more manageable set of values and reduce computation load.



7.1.4 Classification using Supervised Machine Learning

In our given project, instead of relying entirely on one model, we have fitted the model against a variety of classification models in order to obtain the most accurate model and utilize it for our predictions. The models used are Decision Tree, Random Forest, Gradient Boosting, MLP, SVC, K-Neighbour, and TensorFlow Sequential. This is also followed by methodically applying different parameters to the models for further optimization. Through all of this testing, we have found that TensorFlow is the most accurate model and provides us with a final accuracy of 96.45%

7.2 The Application and Graphical User Interface

A simple and fast GUI is also produced separately using Python to provide a straightforward means for the user/customer to use the features of the project with the inner working abstracted from them. The application itself performs prediction on whether or not a spine problem is normal or abnormal and accordingly produces it for the user. It accepts the input data directly from user through editable text boxes and analyzes and fits this data to our optimized model. When the final output classification is produced, it is displayed through a text box to the user for them to acknowledge. The pyinstaller package is used to produce a stand-alone executable of this application.

Chapter 8

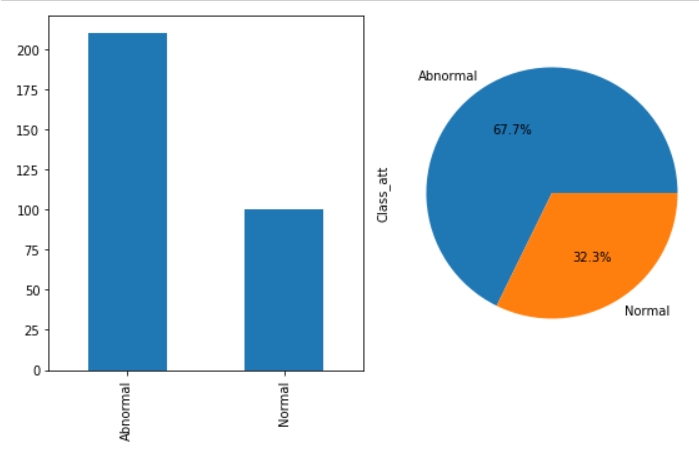
Implementation

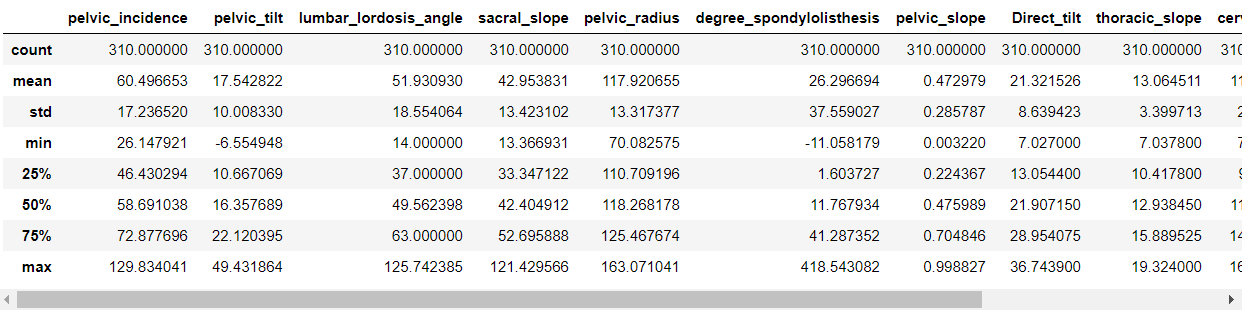
8.1 Exploratory Data Analysis

The libraries of numpy, matplotlib, pandas are imported for mathematical operations on matrices, graphical representation of data and data retrieval and manipulation respectively. The required dataset is imported onto the system. We perform data explorations on the data set, properly label the attribute headers, get summaries, view the range of values of the attributes, check the correlation of each attribute, and so on. After the valuable insights have been extracted from the dataset, we transform the values of the attributes to begin performing operations on them.

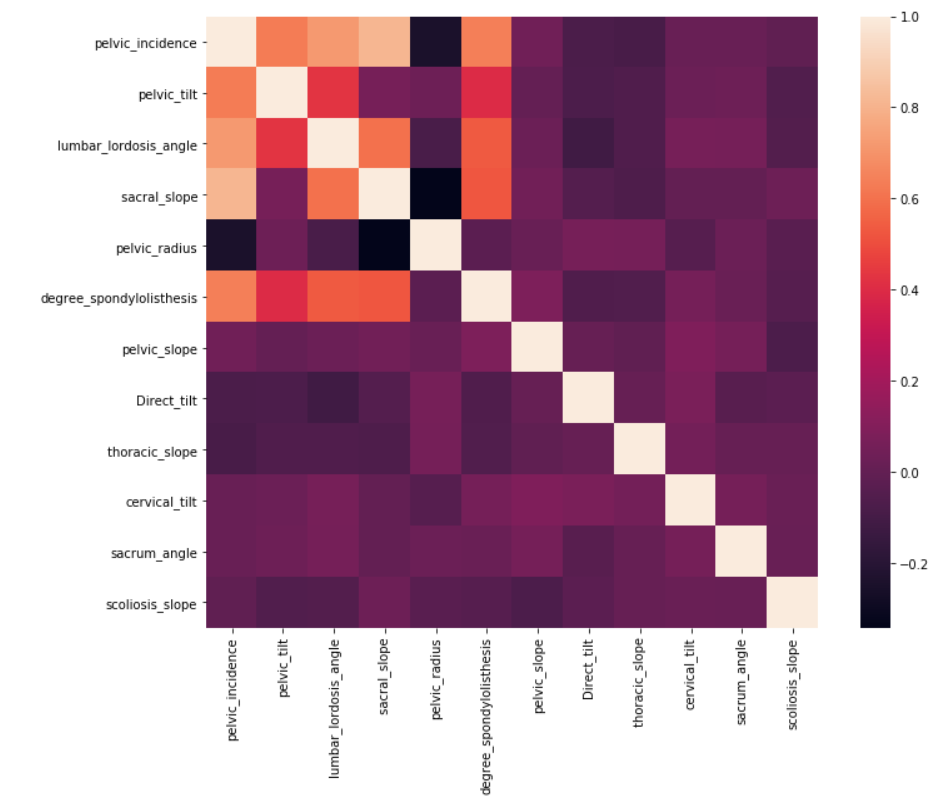
|  |
| --- |
| import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  from sklearn.tree import DecisionTreeClassifier  df=pd.read\_csv('Dataset\_spine.csv')  df.head()  df.columns  df=df.drop(['Unnamed: 13'],axis=1)  df.rename(columns = {'Col1':'pelvic\_incidence', 'Col2':'pelvic\_tilt',  'Col3':'lumbar\_lordosis\_angle', 'Col4':' sacral\_slope',  'Col5':'pelvic\_radius', 'Col6':'degree\_spondylolisthesis',  'Col7':'pelvic\_slope', 'Col8':'Direct\_tilt',  'Col9':' thoracic\_slope', 'Col10':'cervical\_tilt',  'Col11':'sacrum\_angle', 'Col12':'scoliosis\_slope'}, inplace = True) |

|  |
| --- |
| df.describe()  fig, axarr=plt.subplots(nrows=1,ncols=2, figsize=(8,5))  df["Class\_att"].value\_counts().plot(kind="bar",ax=axarr[0])  df["Class\_att"].value\_counts().plot.pie(autopct="%1.1f%%",ax=axarr[1])  plt.tight\_layout()  plt.show()  df.loc[:,'Class\_att'].value\_counts() |





|  |
| --- |
| mat=df.corr()  f, ax= plt.subplots(figsize=(12, 9))  sns.heatmap(mat, vmax=1, square=True); |

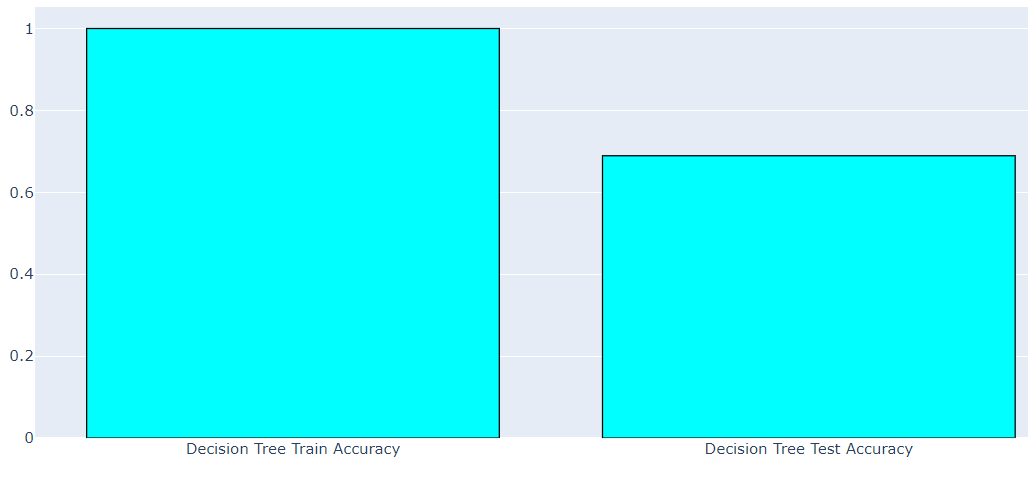


|  |
| --- |
| from plotly.offline import iplot  import plotly.graph\_objs as gph  from sklearn.model\_selection import train\_test\_split  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=42)  from sklearn.preprocessing import MinMaxScaler  scaler = MinMaxScaler()  scaler.fit(X\_train)  X\_train=scaler.transform(X\_train)  X\_test=scaler.transform(X\_test)from plotly.offline import iplot  import plotly.graph\_objs as gph  from sklearn.model\_selection import train\_test\_split  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=42)  from sklearn.preprocessing import MinMaxScaler  scaler = MinMaxScaler()  scaler.fit(X\_train)  X\_train=scaler.transform(X\_train)  X\_test=scaler.transform(X\_test) |

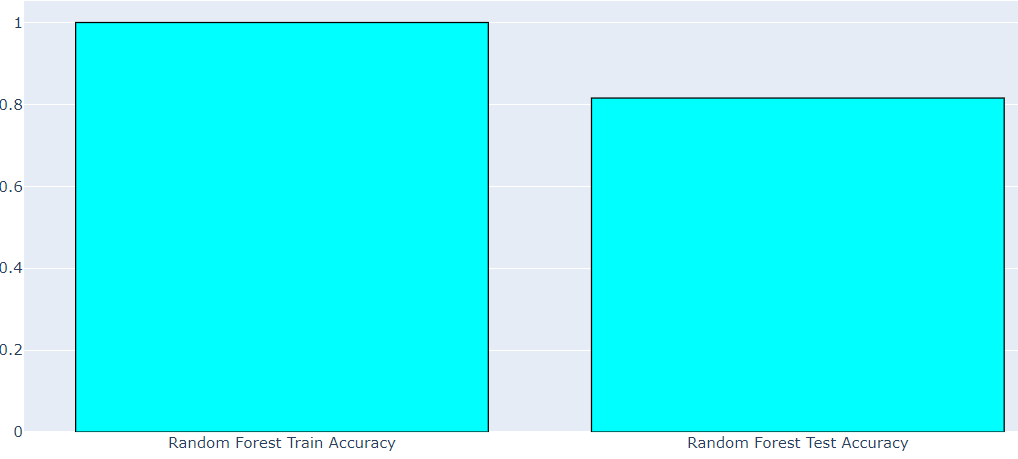
8.2 Model Fitting and Prediction

After obtaining insights from carefully analyzing the dataset and its implications and then transforming this data for finally working upon it, we will begin fitting the data to various classification machine learning models in order:

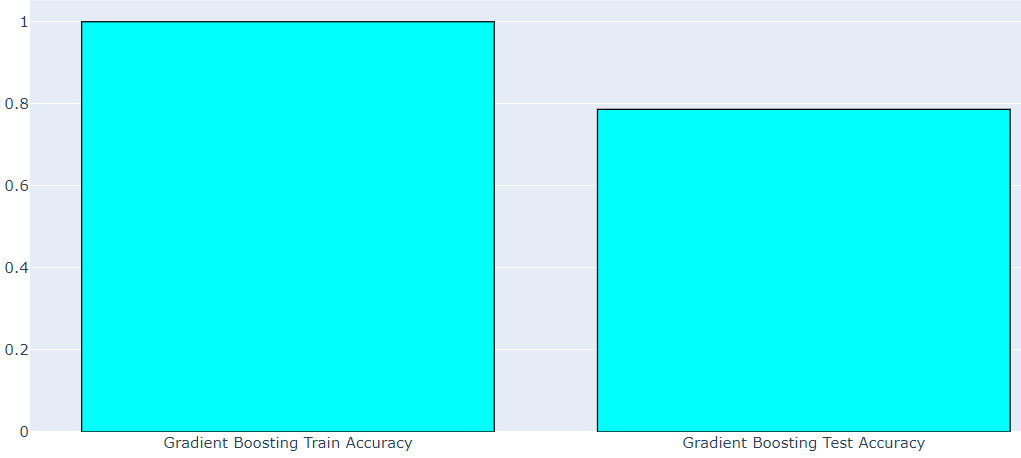
|  |
| --- |
| from sklearn.metrics import accuracy\_score  accuracy\_score(y\_pred, y\_list,normalize=False)/len(y\_test)  Depth = np.arange(1,10)  train\_acc = []  test\_acc = []  for i in range(1,10):  model = DecisionTreeClassifier(random\_state=0,max\_depth=i)  model.fit(X\_train, y\_train)  y\_list=list(y\_train)  print(f"for {i} depth",end=' ')  train\_accuracy= model.score(X\_train,y\_train)  test\_accuracy= model.score(X\_test,y\_test)  train\_acc.append(model.score(X\_train, y\_train))  test\_acc.append(model.score(X\_test, y\_test))  print("Train Accuracy -",model.score(X\_train, y\_train),end=' ')  print("Test Accuracy -",model.score(X\_test, y\_test))  trace1 = gph.Scatter(  x = Depth,  y = train\_acc,  mode = "lines+markers",  name = "Train accuracy",  marker = dict(color = 'green'),  text= "Train accuracy")  # Creating trace2  trace2 = gph.Scatter(  x = Depth,  y = test\_acc,  mode = "lines+markers",  name = "Test accuracy",  marker = dict(color = 'orange'),  text= "Test accuracy")  data = [trace1, trace2]  layout = dict(title = 'Depth vs Accuracy',  xaxis= dict(title= 'Depth',ticklen= 10,zeroline= True)  )  fig = dict(data = data, layout = layout)  iplot(fig)  knn\_train\_accuracy = np.max(train\_accuracy)  knn\_test\_accuracy = np.max(test\_accuracy)  data = [gph.Bar(  x=["Decision Tree Train Accuracy","Decision Tree Test Accuracy"],  y=[model.score(X\_train, y\_train),model.score(X\_test, y\_test)],  marker=dict(color='cyan',  line=dict(color='black',  width=1),  )  )]  iplot(data) |



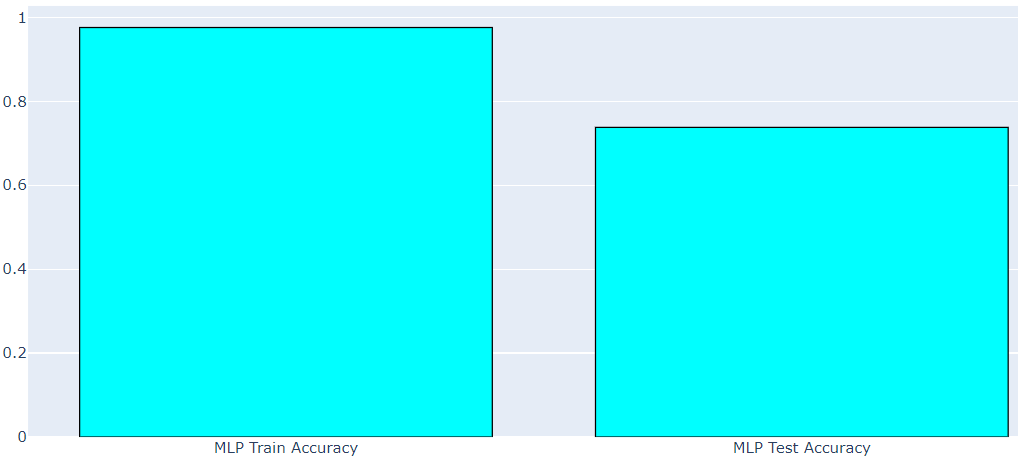
|  |
| --- |
| from sklearn.ensemble import RandomForestClassifier  Depth = np.arange(1,10)  train\_acc = []  test\_acc = []  for i in range(1,10):  model = RandomForestClassifier(random\_state=0,max\_depth=i)  model.fit(X\_train, y\_train);  y\_list=list(y\_train)  train\_acc.append(model.score(X\_train, y\_train))  test\_acc.append(model.score(X\_test, y\_test))  print(f"for {i} depth",end=' ')  train\_acc.append(model.score(X\_train, y\_train))  test\_acc.append(model.score(X\_test, y\_test))  print("Train Accuracy -",model.score(X\_train, y\_train),end=' ')  print("Test Accuracy -",model.score(X\_test, y\_test))  trace1 = gph.Scatter(  x = Depth,  y = train\_acc,  mode = "lines+markers",  name = "Train accuracy",  marker = dict(color = 'green'),  text= "Train accuracy")  # Creating trace2  trace2 = gph.Scatter(  x = Depth,  y = test\_acc,  mode = "lines+markers",  name = "Test accuracy",  marker = dict(color = 'orange'),  text= "Test accuracy")  data = [trace1, trace2]  layout = dict(title = 'Depth vs Accuracy',  xaxis= dict(title= 'Depth',ticklen= 10,zeroline= True)  )  fig = dict(data = data, layout = layout)  iplot(fig)  knn\_train\_accuracy = np.max(train\_accuracy)  knn\_test\_accuracy = np.max(test\_accuracy)  data = [gph.Bar(  x=["Random Forest Train Accuracy","Random Forest Test Accuracy"],  y=[model.score(X\_train, y\_train),model.score(X\_test, y\_test)],  marker=dict(color='cyan',  line=dict(color='black',  width=1),  )  )]  iplot(data) |



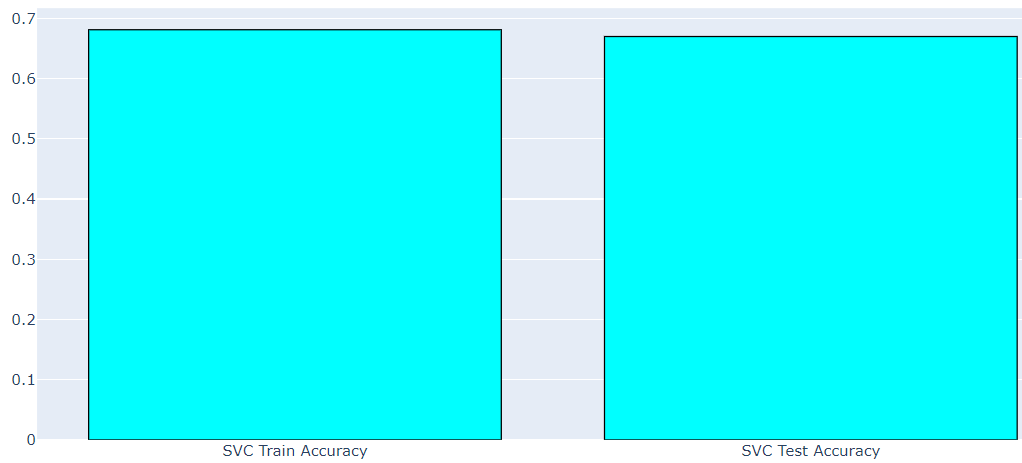
|  |
| --- |
| from sklearn.ensemble import GradientBoostingClassifier  model = GradientBoostingClassifier()  model.fit(X\_train, y\_train);  y\_list=list(y\_train)  y\_pred=model.predict(X\_train)  print("Train Accuracy -",model.score(X\_train, y\_train),end=' ')  print("Test Accuracy -",model.score(X\_test, y\_test))  data = [gph.Bar(  x=["Gradient Boosting Train Accuracy","Gradient Boosting Test Accuracy"],  y=[model.score(X\_train, y\_train),model.score(X\_test, y\_test)],  marker=dict(color='cyan',  line=dict(color='black',  width=1),  )  )]  iplot(data) |



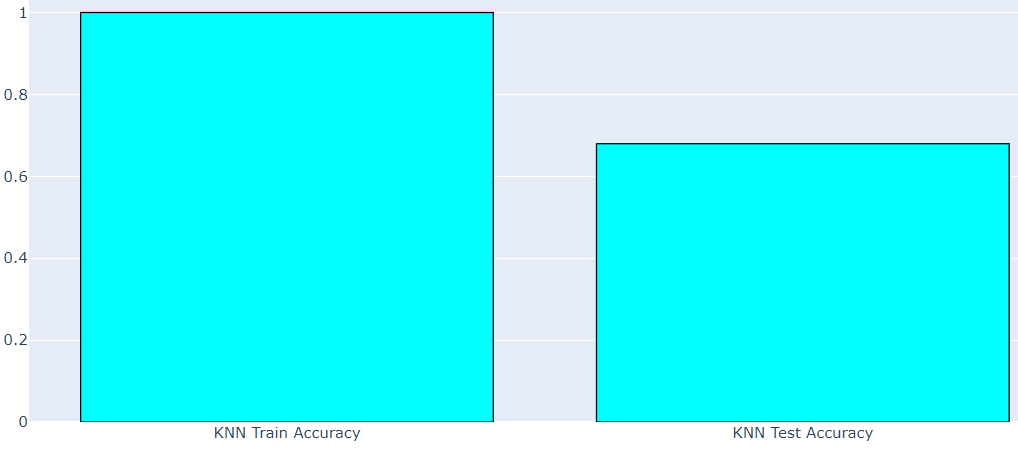
|  |
| --- |
| from sklearn.neural\_network import MLPClassifier  model = MLPClassifier(solver='lbfgs',activation='logistic',max\_iter=100, hidden\_layer\_sizes=(8,6,2), random\_state=1)  model.fit(X\_train, y\_train);  y\_list=list(y\_train)  print("Train Accuracy -",model.score(X\_train, y\_train),end=' ')  print("Test Accuracy -",model.score(X\_test, y\_test))  data = [gph.Bar(  x=["MLP Train Accuracy","MLP Test Accuracy"],  y=[model.score(X\_train, y\_train),model.score(X\_test, y\_test)],  marker=dict(color='cyan',  line=dict(color='black',  width=1),  )  )]  iplot(data) |



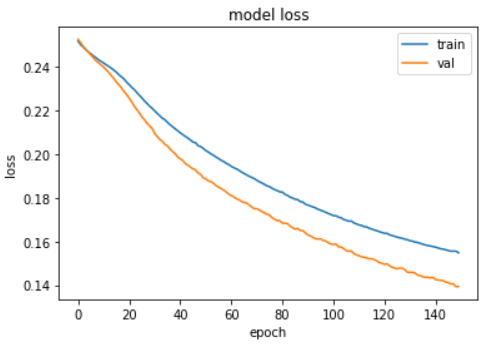
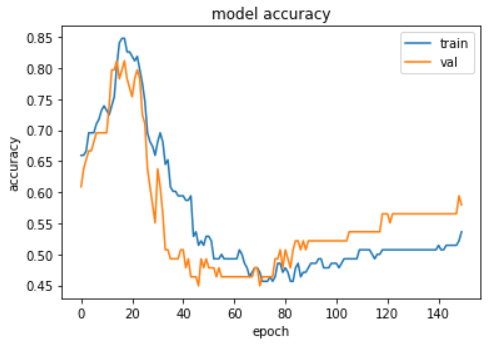
|  |
| --- |
| from sklearn.svm import SVCmodel = SVC(gamma='auto')  model.fit(X\_train, y\_train);  y\_list=list(y\_train)  #print(f"for {i\*0.0001} regularization",end=' ')  print("Train Accuracy -",model.score(X\_train, y\_train),end=' ')  print("Test Accuracy -",model.score(X\_test, y\_test))data = [gph.Bar(  x=["SVC Train Accuracy","SVC Test Accuracy"],  y=[model.score(X\_train, y\_train),model.score(X\_test, y\_test)],  marker=dict(color='cyan',  line=dict(color='black',  width=1),  )  )]  iplot(data) |



|  |
| --- |
| from sklearn.neighbors import KNeighborsClassifier  Neighbor = np.arange(1,15)  train\_accuracy = []  test\_accuracy = []  for i in range(1,16):  model = KNeighborsClassifier(n\_neighbors=i)  model.fit(X\_train, y\_train);  y\_list=list(y\_train)  train\_accuracy.append(model.score(X\_train, y\_train))  test\_accuracy.append(model.score(X\_test, y\_test))  print(f"for {i} neighbour",end=' ')  print("Train Accuracy -",model.score(X\_train, y\_train),end=' ')  print("Test Accuracy -",model.score(X\_test, y\_test))  data = [gph.Bar(  x=["KNN Train Accuracy","KNN Test Accuracy"],  y=[knn\_train\_accuracy,knn\_test\_accuracy],  marker=dict(color='cyan',  line=dict(color='black',  width=1),  )  )]  iplot(data) |



|  |
| --- |
| import tensorflow as tf  import numpy as np  X\_tf=np.array(X\_train)  y\_tf=le.transform(y\_train)  model = tf.keras.models.Sequential([  tf.keras.layers.Dense(4, activation='relu'),  #tf.keras.layers.Dense(6, activation='relu'),  tf.keras.layers.Dense(2, activation='sigmoid')  ])  model.compile(optimizer='adam', loss='mean\_squared\_error', metrics=['accuracy'])  #  history = model.fit(X\_tf, y\_tf, validation\_split=0.33, epochs=150, batch\_size=10, verbose=0)  #model.fit(X\_tf, y\_tf, epochs=100)  model.fit(X\_tf, y\_tf, validation\_split=0.33, epochs=100)  X\_tf=np.array(X\_test)  y\_tf=le.transform(y\_test)  test\_loss, test\_acc = model.evaluate(X\_tf, y\_tf)  print(test\_acc)  import matplotlib.pyplot as plt  plt.plot(history.history['accuracy'])  plt.plot(history.history['val\_accuracy'])  plt.title('model accuracy')  plt.ylabel('accuracy')  plt.xlabel('epoch')  plt.legend(['train', 'val'], loc='upper right')  plt.show()  plt.plot(history.history['loss'])  plt.plot(history.history['val\_loss'])  plt.title('model loss')  plt.ylabel('loss')  plt.xlabel('epoch')  plt.legend(['train', 'val'], loc='upper right')  plt.show()  X\_tf=np.array(X)  y\_tf=le.transform(y)  print(model.evaluate(X\_tf, y\_tf)) |

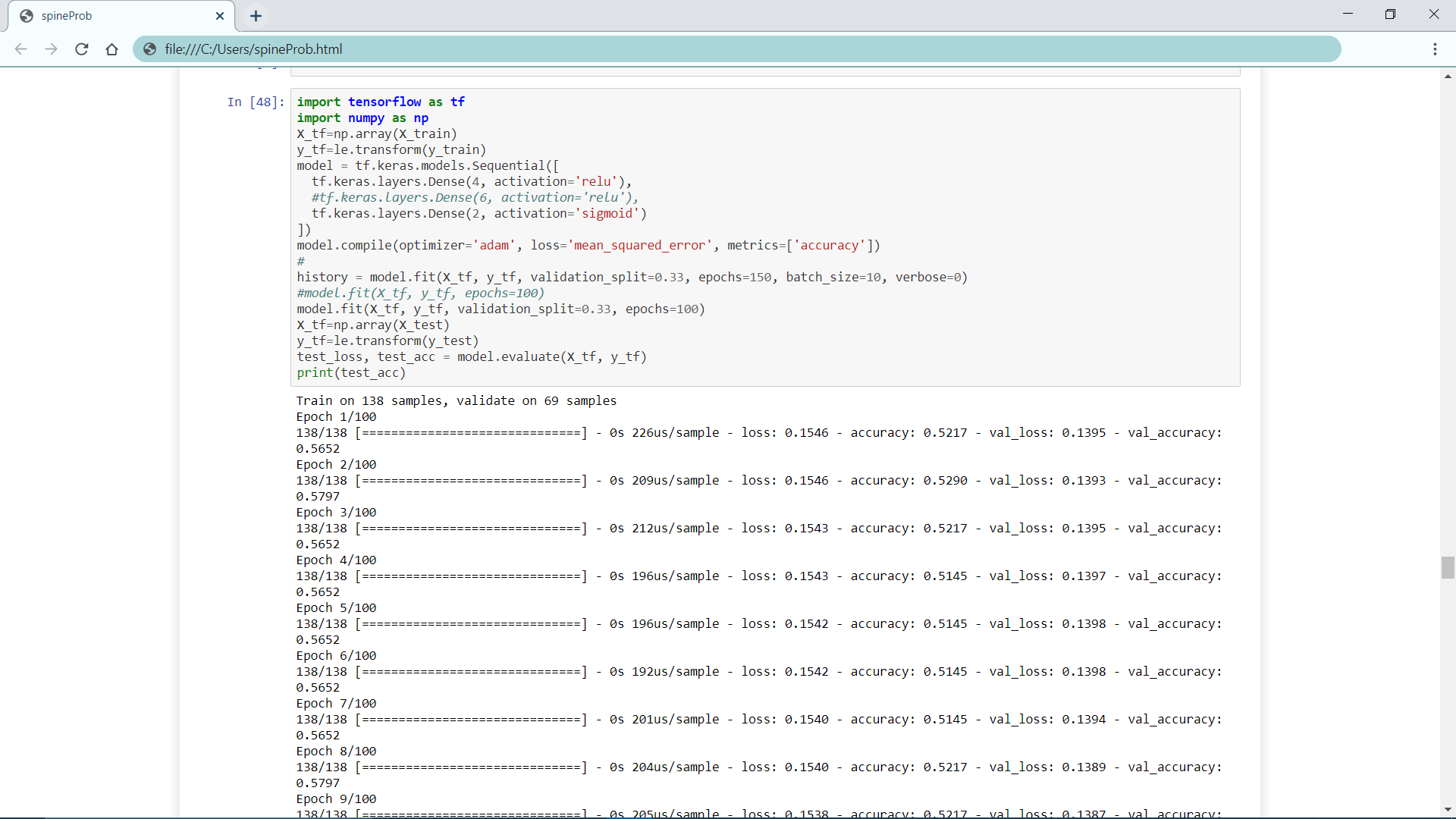


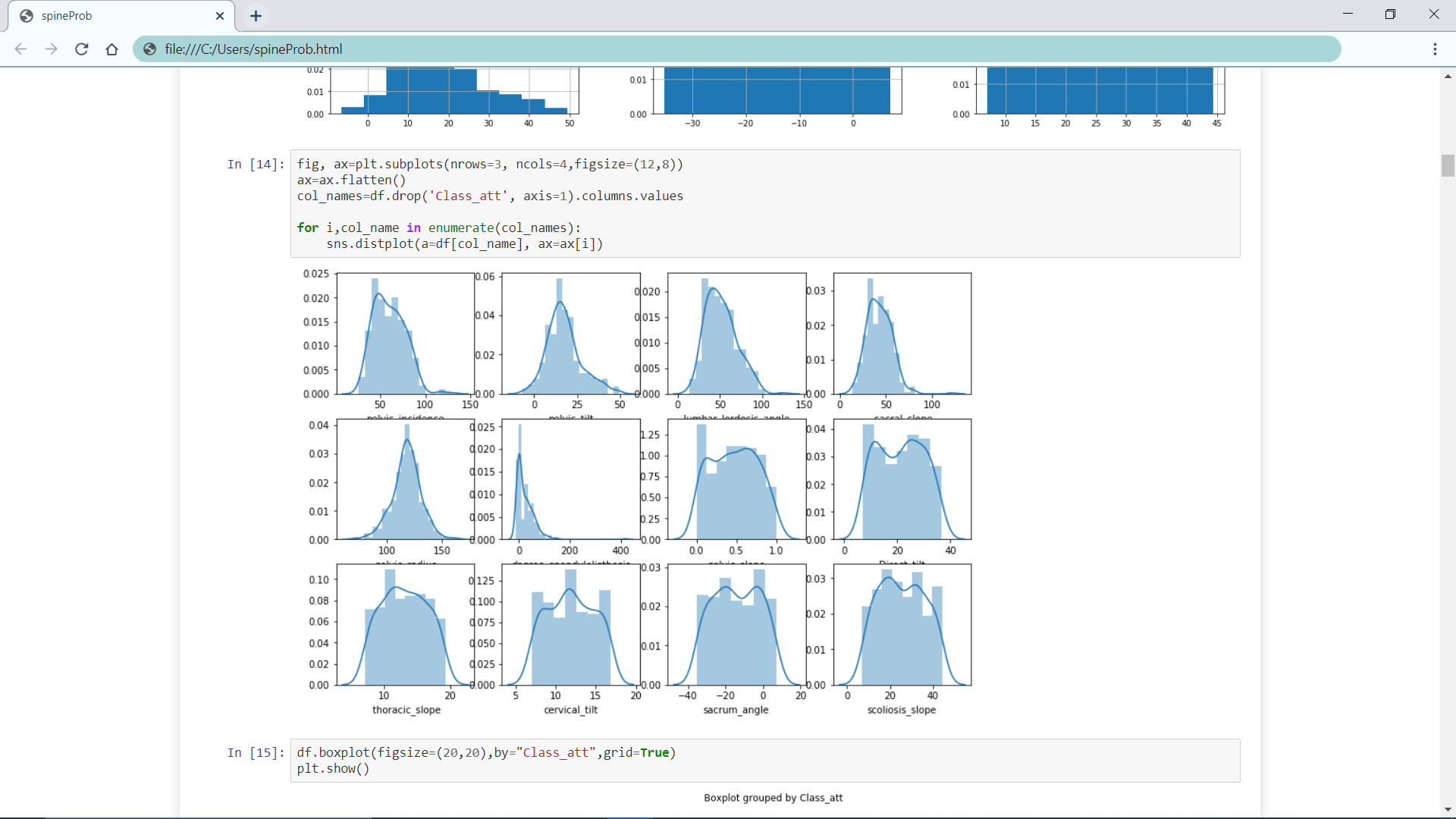
Chapter 9

Screen shots of Project

9.1 Back End

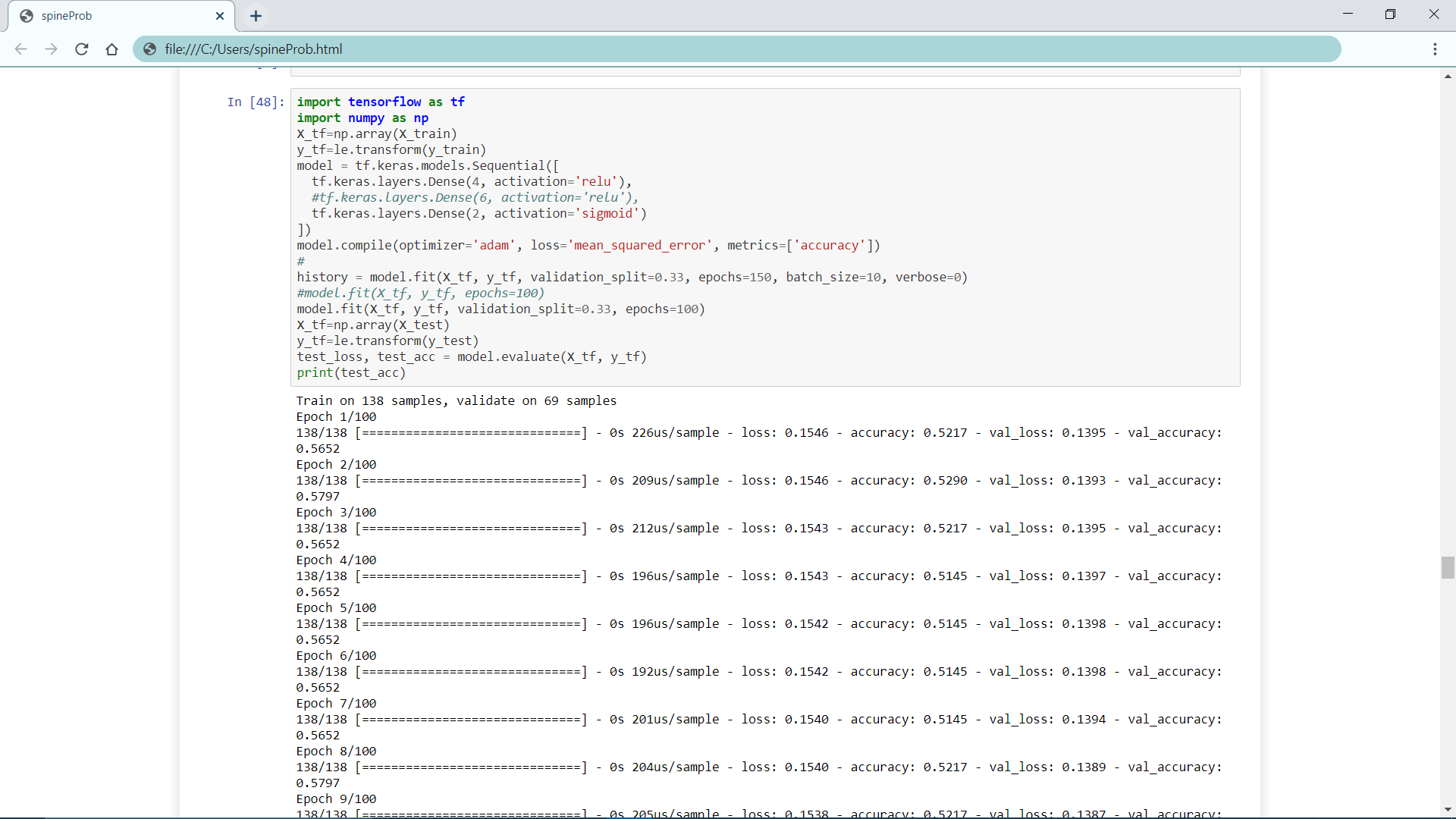
9.1.1 Exploratory Data Analysis



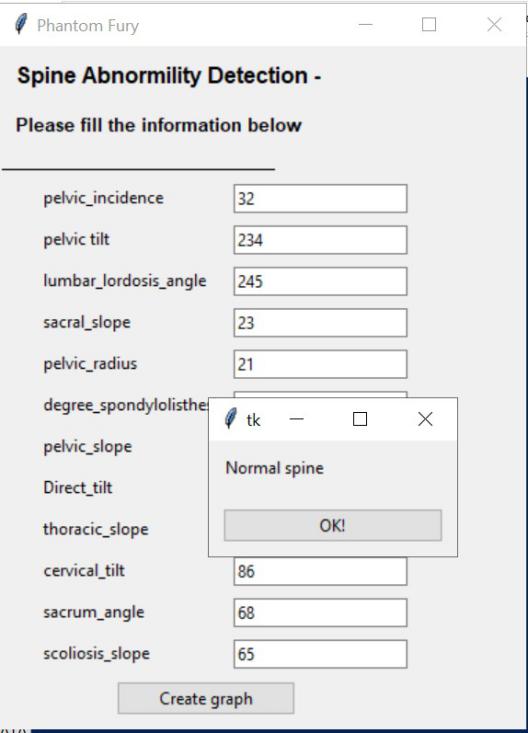


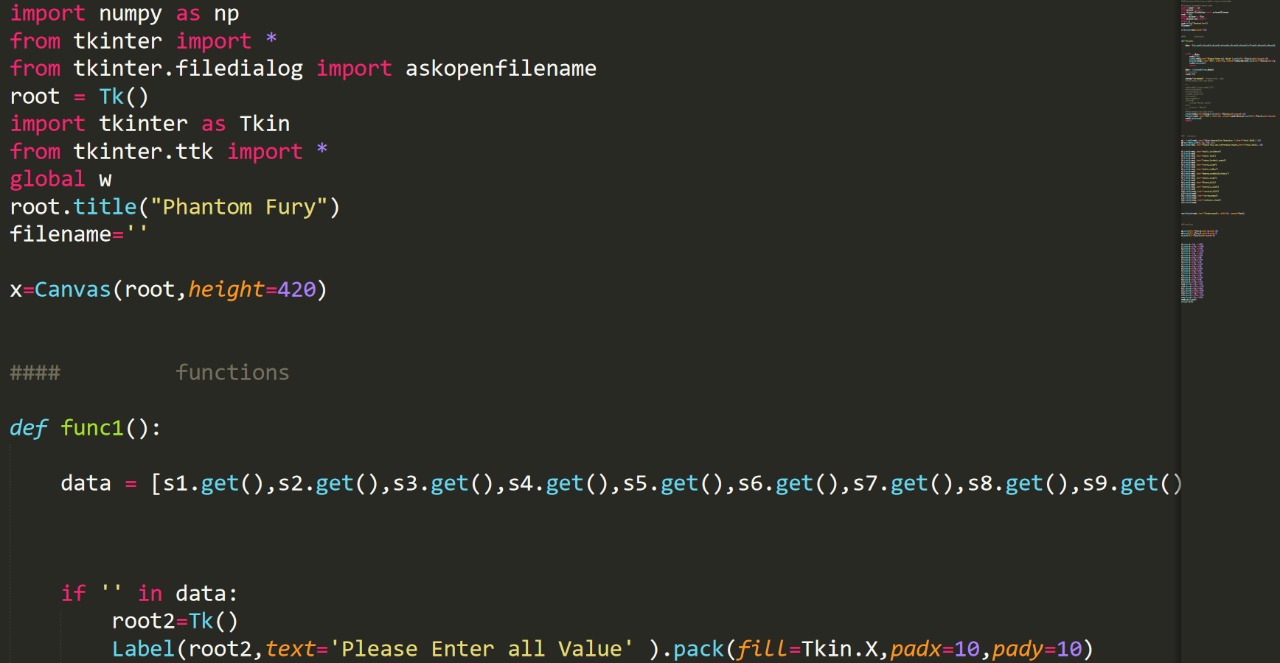
9.1.2 Machine Learning Algorithms





9.2 Front End



Chapter 10

Conclusion and Future Scope

10.1 Conclusion

Finally, we have found that after fitting the spinal biometric data to a multitude of Machine Learning algorithms with varying degrees of success, namely- Decision Tree, Random Forest, Gradient Boosting, MLP, SVC, K-Neighbour, and TensorFlow. Not only do we fit the data to these models but we test the accuracy over a range of values for their parameters to further optimize the results. Finally, TensorFlow Sequential has proven to be the most effective and accurate predictor for this problem statement, hence it will be the final model decided upon to predict the user’s own spinal data and provide insight by classifying it into Normal or Abnormal categories. This interaction between the user and our project is made easier through the use of an application with a very simple and straightforward graphical user interface that can accept biometric data of a patient’s spine and return the prediction after analysis.

10.2 Future Scope

Since our project has successfully predicted the given test set during validation with an accuracy of 96.45% after furthest optimization and model selection, we believe this project can see great use in the industry in the following ways:

* Utilized by any persons who are not so proficient in medical expertise but have performed a preliminary health check up and have access to their details.
* Integrated into a phone application, that can be easily accessed by any individual who can enter their own medical data and get a prediction from the model.
* To be used directly in the medical and health-care industry as a fast and efficient way to diagnose patient with abnormal spinal problems provided their medical details are given.
* Integrated into a larger scope personalized health-care application that can predict a large variety of health issues for patients who are affected by multiple afflictions.

References

1. *Minne H, Leidig G, Wuster C, Siromachkostov L, Baldauf G, Bickel R, Sauer P, Lojen M, R Ziegler .: A newly developed spine deformity index (SDI) to quantitate vertebral crush fractures in patients with osteoporosis. Bone and Mineral 3 335–349 1988 & Maturitas 10(3):248, 1988*

[2] *<https://numpy.org/> *

[3] *<https://pandas.pydata.org/> *

[4] *<https://matplotlib.org/3.1.1/tutorials/introductory/pyplot.html>*

[5] *<https://www.tensorflow.org/>*

[6] *<https://deepai.org/machine-learning-glossary-and-terms/>*

[7] *<https://matplotlib.org/>*

**INDIVIDUAL CONTRIBUTION REPORT:**

**PREDICTING SPINE ABNORMALITIES USING MACHINE LEARNING**

DAIBIK DASGUPTA

1705692

**Abstract:** Due to the long-lasting and painful effects of Lower Back pain caused due to spinal health issues, we have concluded that preventative measures are far superior to curative measures. Following this logic, we have utilized Machine Learning algorithms to accurately predict whether or not a patient is facing abnormal spine problems by analyzing their biometric data.

**Individual contribution and findings:** Comding

**Individual contribution to project report preparation:** Comding

**Individual contribution for project presentation and demonstration:** Presemntation

Full Signature of Supervisor: Full signature of the student:

……………………………. ……………………………..

PLAGIARISM REPORT

