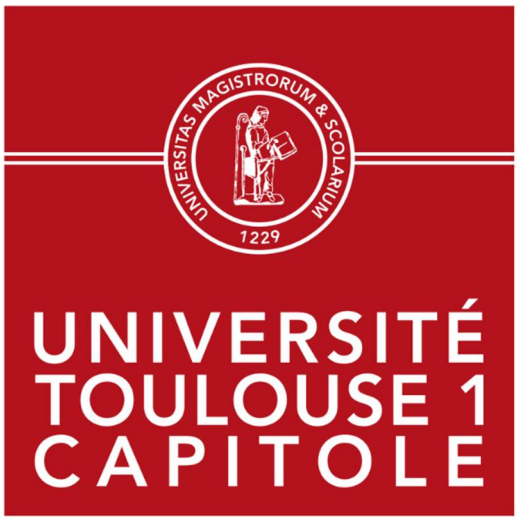


Submitted by

Amrutha Nair THALAPPAN

2IS M1 MIAGE UT1

Student ID - 21911345

ADVANCED PROGRAMMING PROJECT 2: 2019-2020

Technical Specification

Date: Dec-31-2019

Version :1.0

1. Subject description:

This application mainly designed to build a program that creates a decision tree from a given dataset csv file. The application mainly consist of extracting the train.csv file, perform the training based on the characteristics given and predict the target value for the test file. Additionally, the application has a Graphical User Interface to manipulate the functionalities.

2. Major features:

The application must allow the users to choose a character to train the file through the Graphical User Interface. The user can choose characters like 'Passenger class' or 'Sex' from a dropdown list. Based on the chosen characters, the application can train the train.csv file in order to make the application to be predictable when new data comes. Finally, system can predict the target value for a new data set from a file like test.csv in order to map the input characteristic and target value for example predict whether a the passenger is survived or not. Then the predicted result will be shown as a table with passenger id and prediction in the GUI. Moreover I used Tree data structure to implement the decision tree based on the characteristics.

Application allows user to perform below functions:

- Choose a character
- Train a data set by reading train.csv file
- Predict target value for test.csv file
- Export prediction file to another folder location.

3. Data description:

This application has 2 interfaces and 12 classes.

3.1. Interfaces

TrainingValue

An interface to store training values for any character. For e.g.: If the chosen character is Sex, then the training value can be male or female. If the chosen character is Passenger class, then the training character can be 1, 2 or 3. So TrainingValue interface is implemented by two classes, TexualTrainingValue and NumericalTrainingValue. TextualTrainingValue is to store text values and NumericalTrainingValue is for numbers. This interface does not contain any methods or attributes.

ITree

Interface is to create decision tree based on the character chosen. This interface includes all the methods required to structure a tree such as, create node, create edge, update weight of edge etc. This interface is extended by the class Tree. Methods of this interface are

addNode(V v) - To add a new node to the tree

removeNode(V v) - To remove a node from the tree

addEdge(V s, V t, int w) – To add an edge between two nodes with weight of the edge. Here the weight corresponds to total number of people survived or not survived according to the chosen character and initially the weight will be zero. For each passenger we traverse though the tree in order to increment the survival count.

addEdge(int s, int t, int w) – This method is to add precision to the edge.

removeEdge(V s, V t) – This method is to remove edge between two nodes

removeEdge(int s, int t) - This method is to remove precision of nodes of an edge.

isAdjacent(V s, V t) – This method is to check whether the node is adjacent to the other.

isAdjacent(int i, int j) - This method is to check whether the node is adjacent to another in the sense of precision.

getNodes() – This method is to return all nodes in a tree.

neighbors_of(V v) – This method is to return all the neighbor nodes of given node

3.2. Classes

Tree

This class includes method implementations for different operations in decision tree. This class implements ITree. It has two attributes.

Attributes:

nodes – It is list of nodes of

TrainingRow

This class is to store each chosen character and its value corresponds to each row.

Attributes:

rowId - Stores each passenger Id

rowData – Stores data corresponds to each row

TrainingDataSet

This is to create single characteristics tree.

trainingChars – Stores list of characters

trainingRows – Stores data corresponds to each row

TrainingChar

This class is to store the characters which are used to train the data. It has an attribute called 'name' which stores the chosen character.

Attributes:

Name – Stores the choose profile character

TextualTrainingValue

This is to store the training value of each character which has text value. This class has an attribute textValue.

NumericalTrainingValue

This is to store the training value of each character which has numeric value. This class has an attribute numValue.

SingleCharacteristicTree

This class is used to store the result of training.

Attributes:

targerMap – This is a map to store each profile value and its target value. For e.g.: For class 1, target value is 1, i.e. people in class 1 are going to survive.

Methods:

getTargetValue(String profileValue) – It return target value corresponds to the profile value.

toString() – overridden toString() method to print each profile and target value.

DecisionTreeDao

This includes methods to extract Training file and extract testing file respectively.

Methods:

extractTrainingData(ArrayList<String> characteristcs) – It extracts the train.csv file and split the dataset into different training rows using the method setTrainingRows(data) in the trainingDataSet

extractTestData() - It extracts the train.csv file and split the dataset into different training rows using the method setTrainingRows(data) in the trainingDataSet

Attributes:

singleCharacteristicTree – SingleCharacteristicaTree object to call getTargetValue(profileValue) method.

DecisionTree

This is the main class from which the program loads. We directly call the GUI from this class.

Constants

Added all the string constants as a static constants and making use of this class wherever we want.

CSVFileHandler

This class is mainly used for CSV file reading and writing. It includes two methods,

readCSV() – In order to read the csv file and return the data as a Map

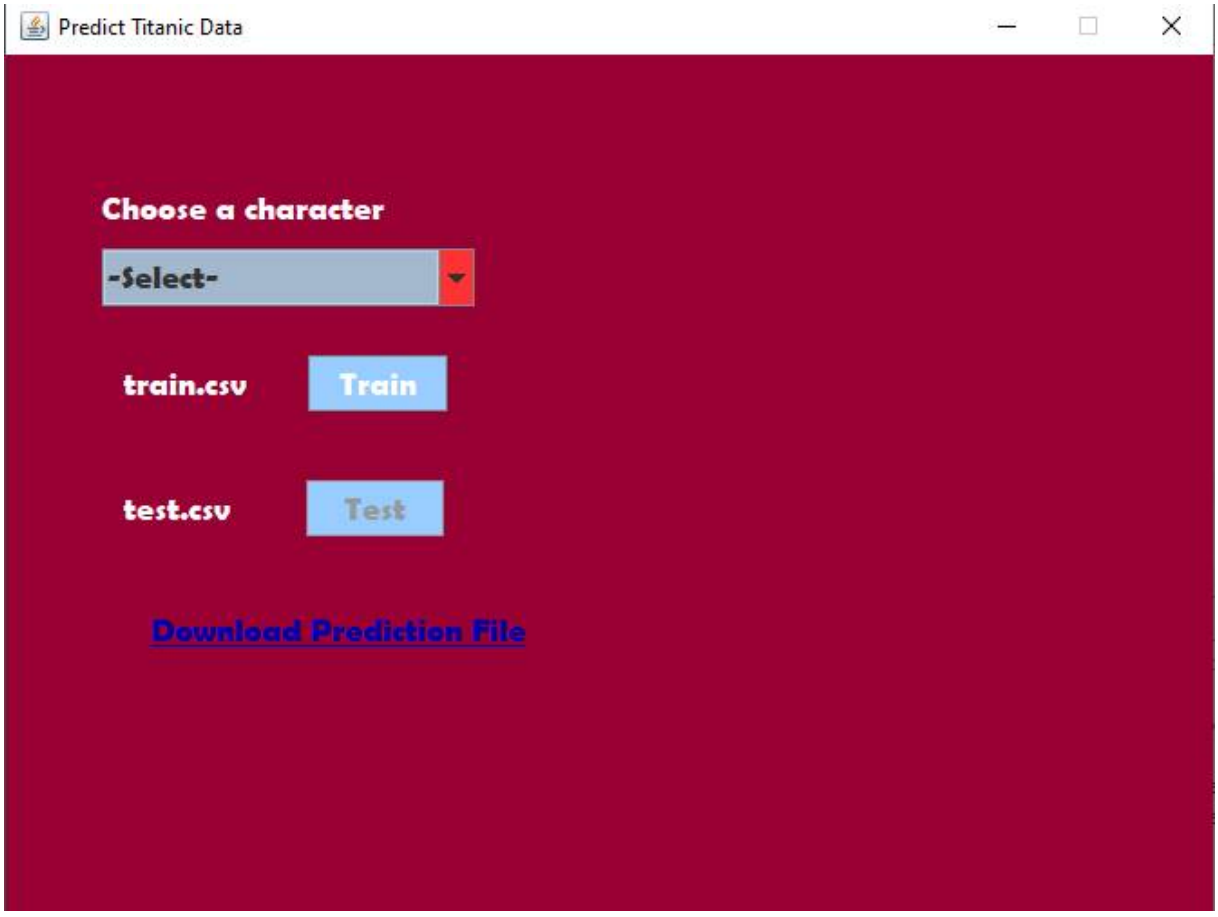
writeCSV() – In order to write the map data to a new csv file.

PredictionUI

This class is the Graphical User Interface of the application.

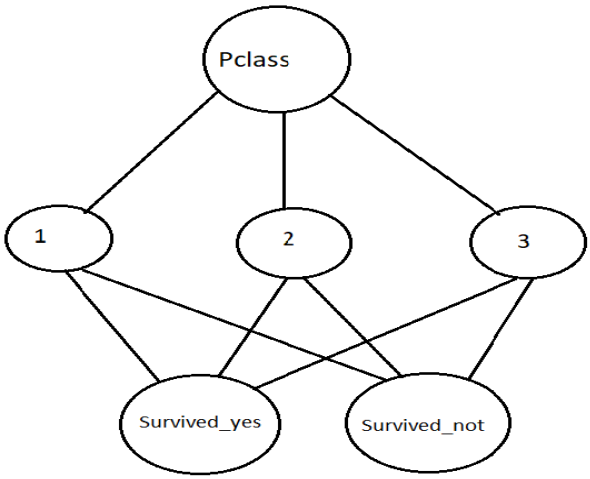
4. Program features

This application designed to make prediction accurately by using decision tree. To make the application more user friendly I created GUI with a dropdown list to choose different characters and a table to the prediction for all passengers. The GUI is shown below.



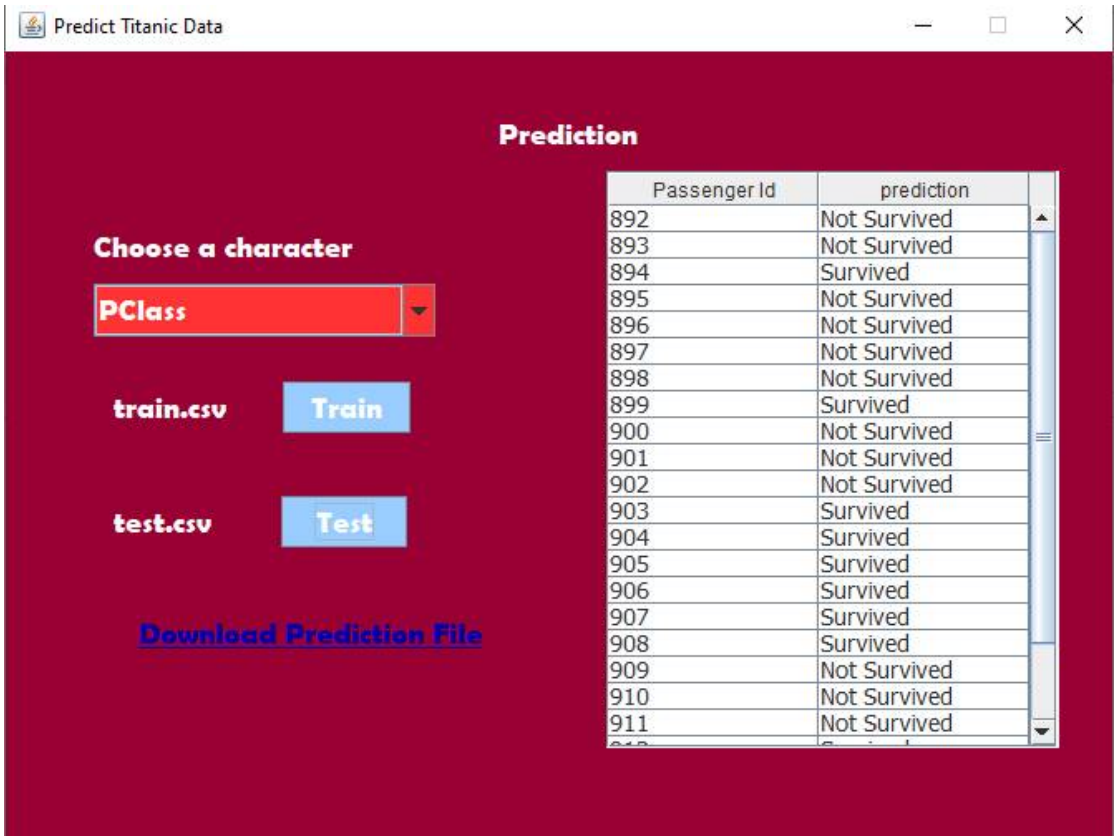
If we click the Train button after selecting the character from dropdown, it will read the train file and make the decision tree based on the values. When you click train button then only the test button will be enabled for further process.

I imported two jars, commons-lang3-3.9.jar and opencsv-5.0.jar for reading and writing CSV file using CSVReadr and CSVWriter. Then when we train the training file, the program will generate decision tree for chosen character. For e.g.: Decision tree for passenger class is shown below.

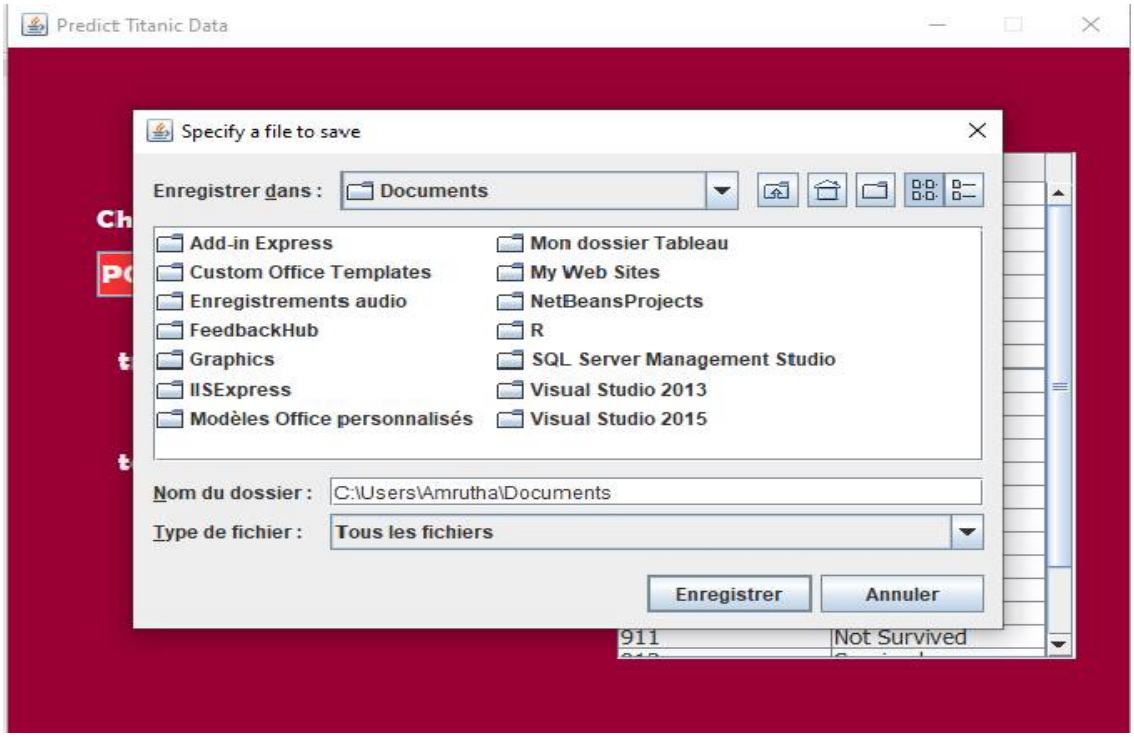


Decision tree making has been done using Tree data structure with an ITree interface. In our application this is a weighted graph. Every edge of the tree has a weight and initially the weight will be zero. But the weight of edges between second level and third level nodes will be incrementing based on whether the passenger is survived or not. For e.g.: If the passenger with class 1 is not survived, then the weight of the edge between the nodes'1' and 'survived_not' will increment. In order to store the weight for each edge I used 'MutablePair' datatype. At the end of the tree traversal we will extract the weight of edge corresponds to the profile and store it into the target map of SingleCharacteristicTree class.

If we click Test button, the application will predict the target values for the test data and it will be displayed in a table with passenger id and corresponding target value. Proper validation for each UI components has been done. The prediction table after testing is shown below.



Moreover, there is a download prediction file link, which is to download entire prediction file containing all details of passengers with survival prediction too. This link will be enabled only after the testing and prediction table has been showed. When user go for training again with any other character then this link and test button will be disabled and get enabled only after corresponding action. The download button action is shown below.



Project build.xml file is modified to accommodate external third part libraries In order to distribute the application software and run as executable file alone.

```
<project name="Titanic_Prediction" default="default" basedir=".">
    <description>Builds, tests, and runs the project Titanic_Prediction.</description>
    <import file="nbproject/build-impl.xml"/>
    <target name="-post-jar">
      <jar jarfile="dist/Titanic_Prediction_application.jar">
        <zipfileset src="${dist.jar}" excludes="META-INF/*" />
        <zipfileset src="lib/commons-lang3-3.9.jar" excludes="META-INF/*" />
        <zipfileset src="lib/opencsv-5.0.jar" excludes="META-INF/*" />
        <manifest>
            <attribute name="Main-Class" value="${main.class}"/>
        </manifest>
      </jar>
</target>
```

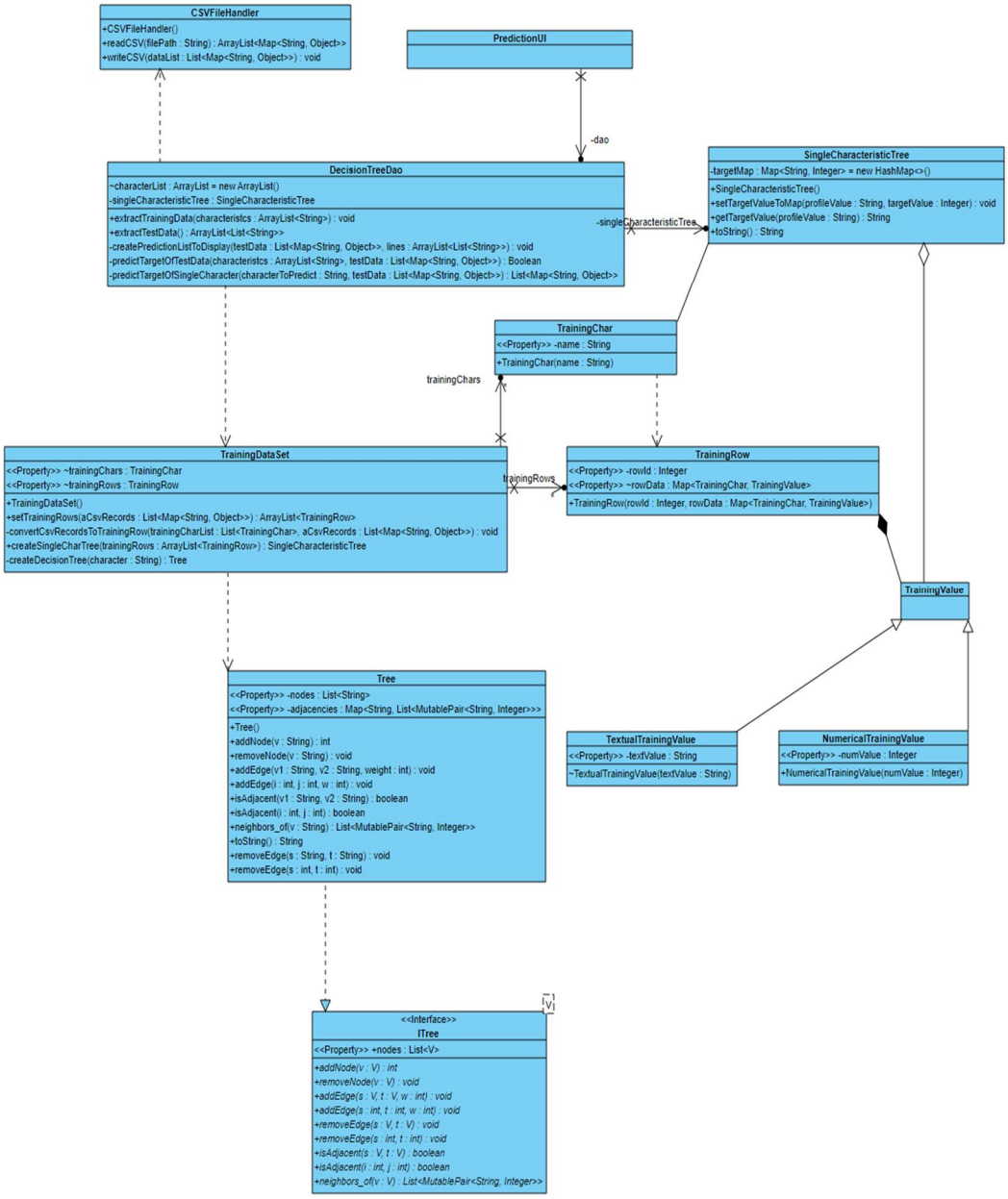
The above changes creates a separate jar files which includes all the external third party libraries used inside the project while performing the build.

5. Scope of the application (limits, evolutions)

- Limits
  - The application do not calculate the accuracy of the prediction
  - It support only two characteristics to train the file.
- Evolutions:
  - Can add multi-characteristic feature to train the file.

6. Class diagram

Based on the program development the class diagram has been changed. New class diagram is shown below,



7. Bibliography

1. http://opencsv.sourceforge.net/
2. https://www.baeldung.com/java-pairs
3. https://docs.oracle.com/javase/tutorial/uiswing/components/filechooser.html
4. https://docs.oracle.com/javase/tutorial/uiswing/components/table.html
5. https://netbeans.org/kb/articles/javase-deploy.html
6. https://m.wikihow.com/Generate-JAR-File-in-Netbeans
7. https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html