The project chosen is the option 1 which is described below.
Option 1: You are working for a non-profit that is recruiting student volunteers to help with Alzheimers patients.
patients. You have been tasked with predicting how suitable a person is for this task by predicting how empathetic he or she is. Using the Young People Survey dataset (https://www.kaggle.com/miroslavsabo/young-people-survey/), predict a person's "empathy" on a scale from 1 to 5.. You can use any of the other attributes in the dataset to make this prediction.

Firstly, the data files are read into the notebook. The dataset contains 150 features mentioned in the columns.csv file.
The data given is split into train, development and test set by using train_test_split function twice. So the first split produces train of 80% and 20% test. Again train data is split using same function in same ratio resulting in 64% of data being train data and 16% being development data. So these splits give train of 64%, development of 16% and test data of 20%.

For pre-processing, null valued columns are replaced with the respective modes(majority value). First I tried to run classifiers on this data and it gave me errors as some of the columns were not numeric. The dataset contains some non-numeric datatypes ie, object.

These attributes are
namely-'Smoking','Alcohol','Punctuality','Lying','Internetusage','Gender','Left - right handed','Education','Only child','Village - town','House - block of flats'], prefix=["sm", "al", "pu","ly","internet", "gender","handedness","edu","onlychild","town","house".
First I tried to give cat codes but it is said that it gives undue advantage to higher codes. Therefore I tried using one hot encoding.

The object columns are converted to categorical datatype and then they are converted to numeric using one hot encoding where in we introduce a new column for each of the attributes values. This is done using pandas get_dummies function.

Since Empathy has to be predicted. First Empathy column is extracted and it is stored in y variable(the label). The rest of the data is put in the X variable. Then after splitting the data.

Training data is used and decision tree is used which gave test accuracy of 0.321782178218. With naive bayesian classifier test accuracy was 0.351485148515.

Later on Principal Component Analysis was tried with features being limited to 20. This gave an development accuracy of 0.33950617284.For 30 components the development accuracy was 0.358024691358.
So 30 components were chosen which gave a test accuracy of 0.391188118812 .
The PCA was chosen because there were too many features. When there are too many features, eliminating features that are not correlated with the label gives us better accuracies. This we can see in the above values.The PCA method improved accuracy by at least by 4%. Usually we evaluate success by comparing our test accuracies with others who tried the same problem. This is because the accuracies depend on the dataset given. Sometimes the there may not be good accuracies for the dataset.

The coding is done in jupyter notebook. The packages used are pandas, numpy, sklearn, sklearn.model_selection, sklearn.decomposition, sklearn.linear_model.
Code link can be found at https://bitbucket.org/niharikadharmapuri/imlhw5