Main steps in the project.

Step 1: Visualization of data

- Pie chats were generated for categorical data : *1_DataVisualization.pdf* (*F*igures 1-8)
- Stacked Bar charts were generated by grouping each categorical data with "class" (i. e. <=50K, >50K) , for every categorical attribute another set of Scaled stack chart was generated. : :*1_DataVisualization.pdf* (*F*igures 9-24)
    - The stacked bar chart gives a visualization of the distribution of class within each attribute.
    - The Scaled stacked bar chart allows better visualization of the percentage distribution of class wihin each sttribute.
- Each continuous attribute was discretized by different bin width (3 sizes of bins were used for each attribute) . Histogram and Scaled bar chart were generated for each attribute with different bin width : *1_DataVisualization.pdf* (*F*igures 25 -60)

<u>Discussion:</u>

Observation of the data by visualization technique highlight some of the general characteristics of the data:

1. People with higher education seems to more likely have salary >50K (figure-12). The stacked bar chart shows that people having , masters, doctorate or Professional education are more likely to earn >50K. Education-number indicates the similar trend, i. e higher the education more the earning.
2. Married people are more likely to have earning >50K, this could raise two possibilities/ questions:
    a. Married people strive to earn more or tend to find higher paying job?
    b. People who earn more, are more likely to get married or remain married?
3. Males are more likely to have earning >50K as compared to females.
4. Further people with certain types of occupation like " exec managers", "prof-speciality" , are more likely to earn higher than occupations like "private house ser", " handlers-cleaners"
5. Some attributes like "fnlwgt", "race" do not appear to show much impact on decisiveness when data is observed visually, although to prove this observation, further experimentation is needed.

Step 2: Dealing with missing values and Data Preprocessing

- Missing Values that were represented in data by "?" were noticed only in three attributes:
  "native-country", "workclass" and "occupation".
- None of the continuous variables showed missing values.

Two approaches that I took to handle the missing values:

1) **Drop the missing values :**
   - The records that had one value missing, had one or more of other values missing as well.
   - Further there were > 54000 labelled records, thus after dropping the missing row, we will still be left with ample labelled data to train the classifier.
     -

2) **Replace the missing values with central values / most frequent values:**
   - Since none of the class labels are missing, I grouped the data based on the class labels (>50K, <=50K), and used central value interpolation (mean/ median) within the group to replace continuous data (if any missing), and used mode within the group to replace the missing value in categorical data.

Method in module *preprocess.py*

`dealMissing(df, dataType, choice=1),`

- `takes as input the`
  - `df: data ,`
  - `datatype : an array that indicates which variable is categorical and which is continuous`
  - `choice`
- `Drops missing value records if choice is 1`
- `Replaces missing values with median/mean/ mode, if choice is 2: Here the method` **`checks if the continuous data is normally distributed`**`, if yes it replaces with mean. If is data is not normally distributed, median is used to replace the missing value. For categorical data mode is used to replace the missing value.`

Step 3: Implementation of Naïve Bayesian Classifier

Discretization of Continuous Data: (preprocess for NB classifier – A)

From module ***preprocess.py*** –method: `def makediscrete3(df, dataType):` `takes the dataframe (df) and datatype dictionary ( that tells the data type) and discretizes continuos data based on the number of bins given by user for each attribute. The method supports creating equal sized bins / customized variable sized bins.`

A: Discretize continuous data approach (needs discretized continuous data)

- Method **`naiveBaysianClassifier`**`(training, test, dalaLabels), takes the`
  - `Training: dataset in the form of 2d array`
  - `Test : dataset in form of 2d array`
  - `DataLabels: a dictionary to indicate categorical/ continuous variables.`

Used training set to trainin the classifier and test dataset to evaluate , and writes the output to the file.

- Method **`trainingNBcat`**`(newData):`
  - Takes the newData (training dataset), and trains the classifier, returns the dictionary of **Likelihood** P(x|c) (the probability of *predictor(attribute)* given *class)*, and also returns the probability of the class (**Class Prior Probability**) P(c) .
  - Lapacian correction is used to handle the zeroing of the probabilities.
- Method testNb():
  - Takes one record from test dataset and calls method `classify(onerow, probDict, classCount, classLabels)`
  - Method classify () takes one record and calculates P(c|x) – probability of class given x , based on likehood and prior class probability, checks the max P(c|x) and returns the class label. Thus, this method classifies the data.
  - The returned value from classify() is tested against true label to make a confusion matrix and calculate accuracy. The accuracy and confusion matrix is returned.
- **A: Guassian approach**

Guassian implementation of the Naïve Bayesian classifier is done in module ***GaussianNB.py***

- Method `def NBGuassian(training, test, dalaLabels):` Used training set to train the classifier and test dataset to evaluate , and writes the output to the file
- `def trainingNBcon(newData):`
  - Method trains the classifier and returns the dictionary of likelihood for categorical variables and (mean, standar Dev) for continuous variable.

- `def testGuassNB(newData, probDict, classCount, classLabels,  myclass='class'):`
    - Method applies the classification to the test dataset and returns the accuracy and confusion matrix.
- `def GuassClassify(onedata, prob_list, classCount, labels)`
    - Method uses the dictionary returned from **`trainingNBcon`** to test the Record.
        - For continuous variables mean and standard deviation is used to calculates P(c|x) by calling : `def calculateProb(x, mean, stdev):`
        - For Categorical data : P(c|x) is calculated likelihood from prob_list- the dictionary returned from trainingNBCon.

Step 4: K Fold Evaluation

- Method in Module ***KFoldEval.py***: `def kfoldEvaluation(df,dalaLabels, classifier, k=10):`

  Takes the dataframe (df) and divides it into 10 parts , picks one part as test set and used 9 other parts as training set for the classifier that is passed. The default k=10, other values of k are supported by argument.

- Method `evaluateNB(labeledDataF, dalaLabels, choice=1):` Calls the k fold evaluation, and passes different classifiers , writes the output to the file.

Discussion

Findings:

- The Average accuracy of Classifier ranged from 80.29 -83.26
- The Best accuracy was found when missing values were replaces (NOT DROPPED) and continuous data was handled using Gaussian formula.
- Among the discretization results, best accuracy was obtained when custom bin sizes were used.
- When equal bin sizes were used 3 equal bins gave better accuracy that 6 equal bins.

	Missing values dropped		**Missing Values Replaced**	
run specifications ▼	**DicreteNB (mean ± std)**	**GuassianNB (mean ± std)**	**DiscreteNB (mean ± std)**	**GuassianNB (mean ± std)**
Equal bin sizes : 3	81.10 ± 0.47	82.58 ± 0.33	81.65±0.53	83.26± 0.33
Equal bin sizes : 6	80.29 ± 0.50	82.58 ± 0.33	80.92 ± 0.64	83.26± 0.33
Different Bin Sizes: age : equal 3 bins fnlwgt :equal 3bins education-num: 3 ustom bins [0-8, 8-12, 12-16] capital-gain: 2 custom bins [0-5000, 5000 -onwards] capital-loss : 2 custom bins [0-4000, 4000-onwards] hrs-per-week: 3 bins [0-30, 30-50, 50-onwards]	81.54 ± 0.49	82.58 ± 0.33	81.99 ± 0.51	83.26± 0.33

Discussion:

Based on the results of my experiment, dropping the records with missing values resulted in loss of accuracy. This indicates that count of labelled data is important in classifiers accuracy and attempt should be made to replace the missing value whenever appropriate, instead of simply dropping the records.

Further customized bin discretization improved the accuracy of the classifier. Bin sizes and ranges should be customized according to the labelled data. Data visualization is thus an important step to indicate the proper bin sizes and ranges.