Subarna Chowdhury Soma
SJSU ID: 014549587

1. ***(a) [10 points]***

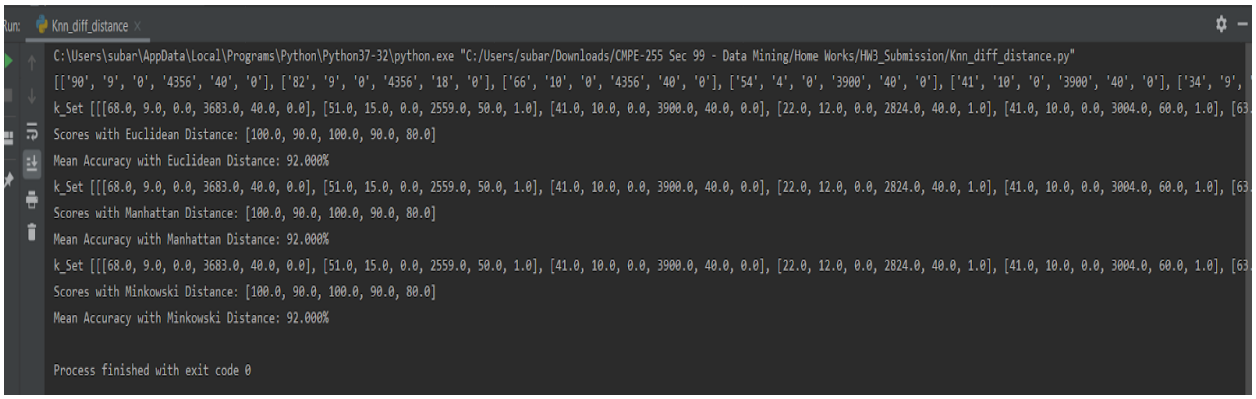
Refer to online tutorials on K-NN implementation such as https://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/

.Extend the implementation to use various distance metrics such as Manhattan distance and note if the classification changes with the distance metric (for a more exhaustive list of distances, see getDistMethods() in R). Use the same dataset as you used for HW2, Q2b.

Answer:

File: hw3_knn_with_diff_distance

Output and Observation:



I have implemented five distance metrics for knn algorithm on adult_Census_income dataset from Kaggle [HW2-Q2b]. Implemented distance metrics are

1. Euclidean Distance,
2. Manhattan Distance
3. Minkowski Distance
4. Cosine Distance
5. Mahalanobis Distance

I have found that accuracy is the same for all the calculated type of distances except for cosine distance. I have calculated and evaluated algorithm with different K value and finally used the optimal k value.

(b) [10 points]

In K-NN, we ignored the direction component of the vectorized representation of data items and only considered the distance. Does it make sense to also consider the direction of the nearest neighbor in addition to or instead of the distance from it? Why or why not?

Answer:

Yes, we can also consider direction component as a measure while implementing K-NN algorithm. Direction is measured by cosine similarity. Cosine Similarity is formulated as to measure the difference of angle between two vectors. As length of vector is not considered here, we can also normalize each vector to a unit vector first and then calculate the inner product of two vectors.

So cosine similarity can be measured between two vectors A and B to calculate direction for K-NN using the following formula:

Sim(A ,B) = Cosθ = A.B / (|A||B|) , where θ is the angle between vector A and B.

So direction can be considered for nearest neighbor calculation instead of different distance matrices. But sometimes, cosine similarity can not be used a proper distance calculation metrics because of the triangle inequality violation.

But there are some situations where we can consider direction and/or distance for K-NN algorithm based on dataset. There are also some cases where direction will not give good accuracy for K-NN. Following are some cases to consider:

Case 1: When data points are moving [e.g. road network], considering direction with distance for K-NN algorithm will give high performance. That's why direction-aware K-NN algorithm is being used for road network.

Case 2: Let's consider highly sparse data set, where two vectors are rarely similar. The, if a data point has no neighbors among potential candidates of nearest neighbor, of the same features, all the cosine similarity becomes and it will be difficult to find the nearest neighbors. Again for point like, (1,1) (2,2) direction is same. For both of the cases considering direction for K-NN will not be effective.

In short, we can conclude that like no-free-lunch theorem, no optimal distance metrics/direction can be used for all types of dataset. It is obvious and clear that the cosine distance(direction) is not overall best performance metrics and performs worst

on sparse noisy data set. Some recent research shows that, for some data sets [e.g IRIS data set of 1(a)] , direction outperforms other distance metrics for K-NN algorithm.

2. (a) [10 points]

Go through the data preprocessing tutorial at https://www.kaggle.com/wencesvillegas/titanic-dataset-preprocessing

List out as many different alternate schemes of preprocessing as you can for the same Titanic dataset, which will result in better performance. For instance, the tutorial assigns 'Unknown' to the missing cabin codes. Instead, we can decipher the cabin code based on last name. E.g.: If Mrs. Cumings' cabin code is 'C85' and Mr. Cummings' cabin code is missing, we set the latter to 'C85' as well. Change the code to implement a few such improvements to see if the strategy works. The dataset can be downloaded from https://www.kaggle.com/c/titanic/data

Answer:

File 1: hw3_preprocessing_titanic_data_tutorial

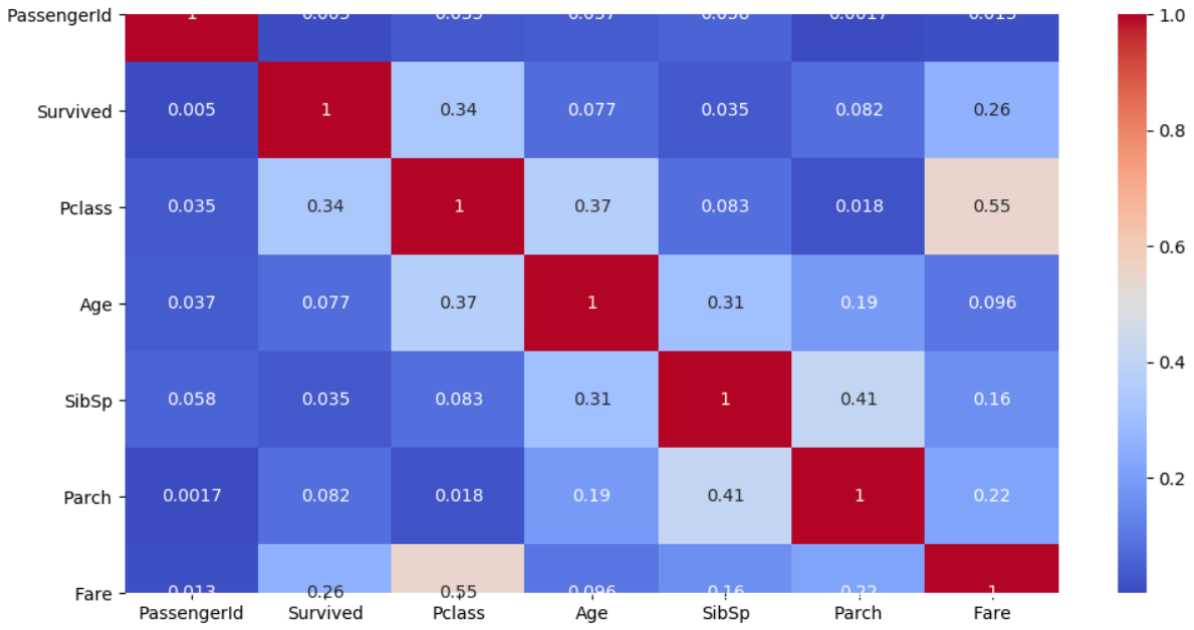
File 2: hw3_new_preprocessing_titanic_data

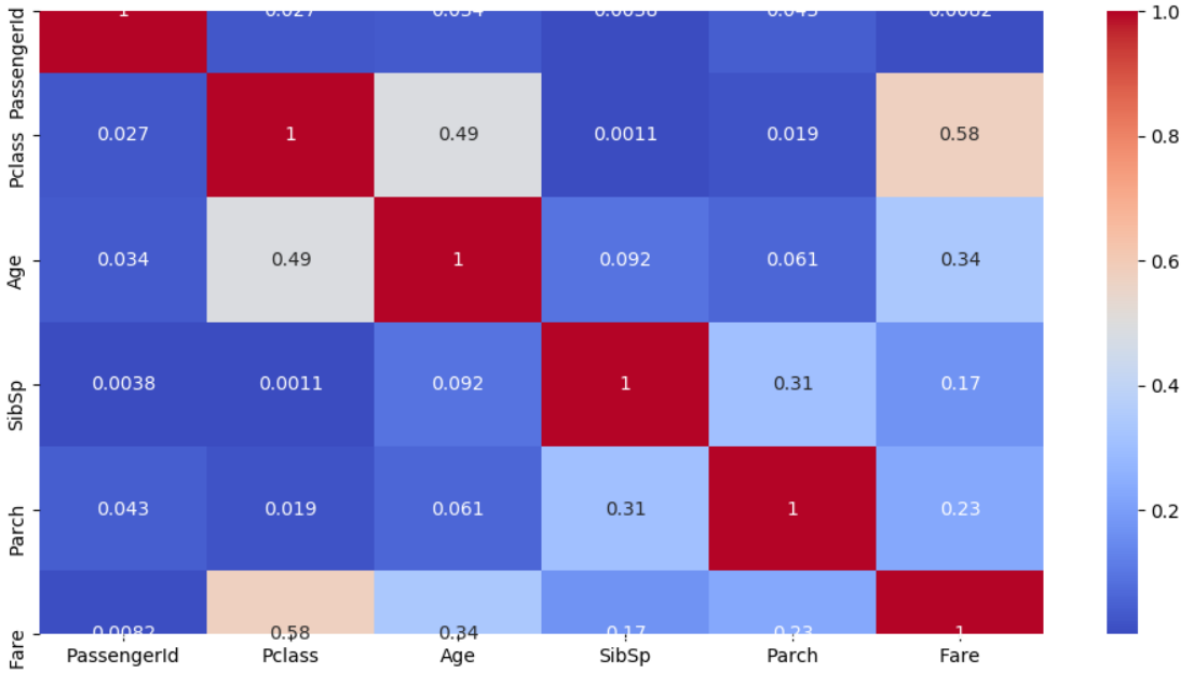
Given tutorial has applied the following preprocessing techniques:

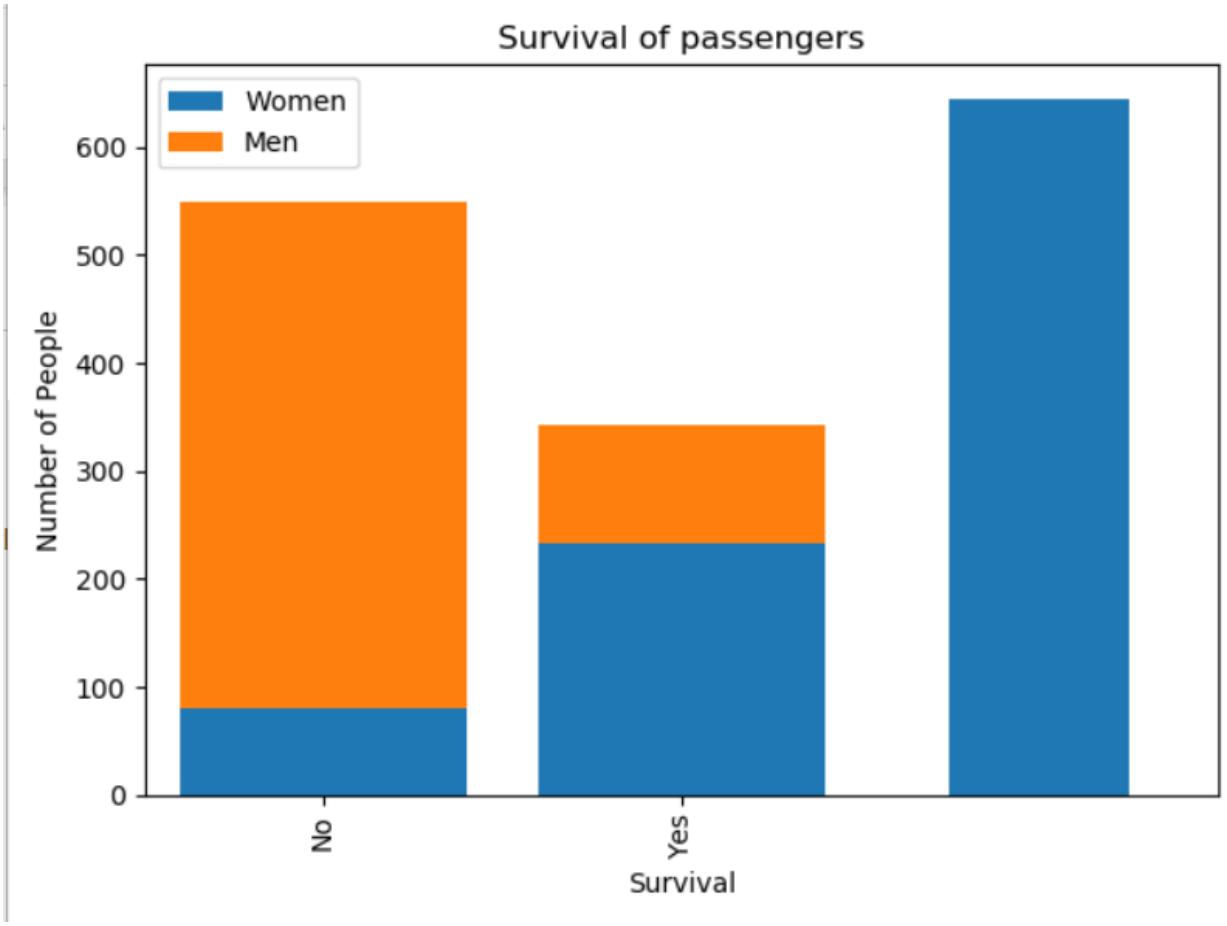
a. Convert Cabin feature to Deck feature
b. replace passenger's name with his/her title (Mr, Mrs, Miss, Master)
c. Delete Cabin and Name feature
d. Dropping less insightful Ticket column
e. Replace each NaN value with the mean of the age of the corresponding person's title
f. Replacing NaN of Embarked column with most common value 'S'
g. Adding new feature 'Family Size' as a sum of two similar features 'SibSp' and 'Parch'
h. Standardizing numerical feature Age and Fare for missing value
i. Encoding categorical classes (LabelEncoding)
j. Encoding categorical features such as Embarked, Title and Pclass by getting dummies
k. Cleaning Test set by aligning data frames to fill Deck column

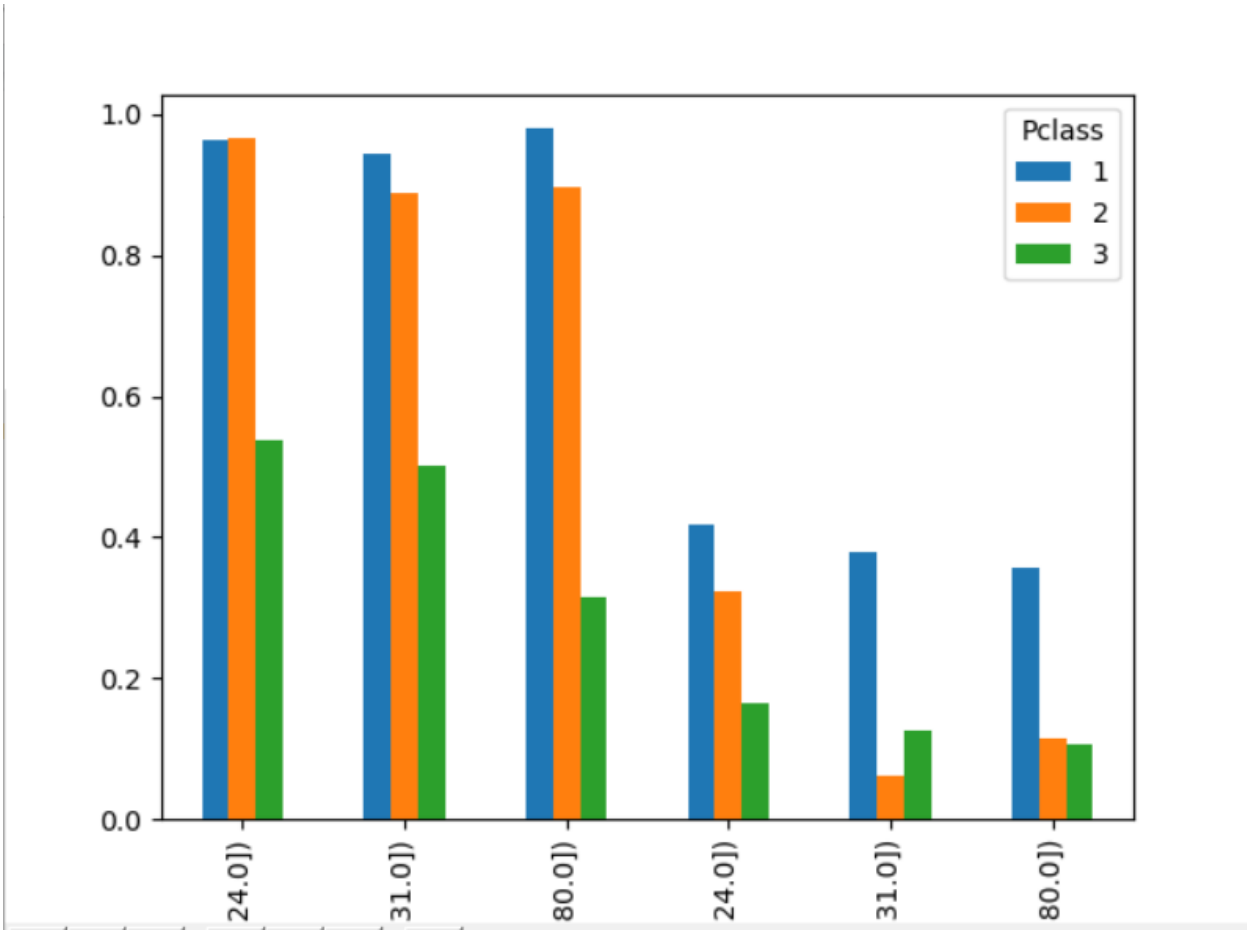
My Processing which are different from given tutorial:

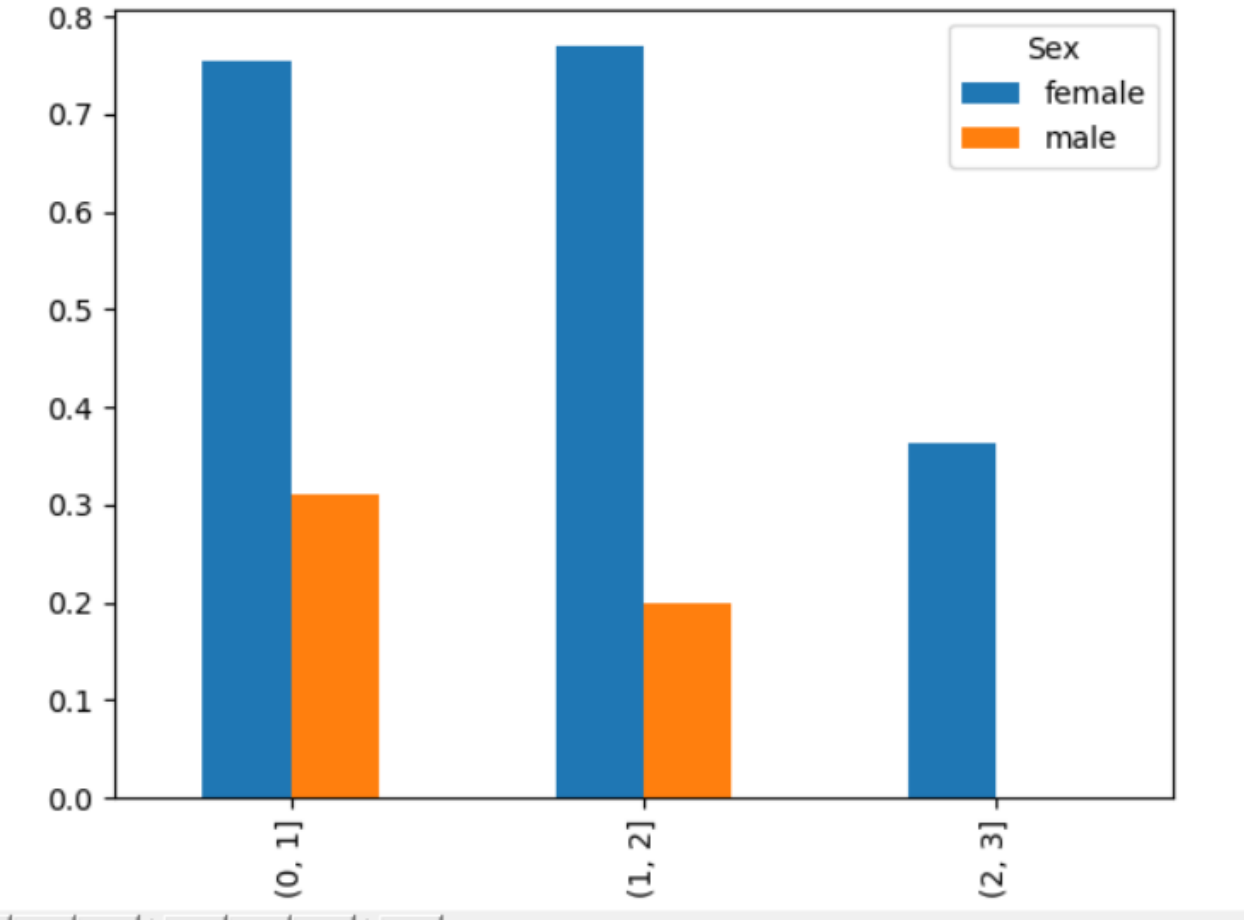
a. Dropping columns with less priority, e.g. Ticket
b. Working on missing valued columns ‘Cabin’, ‘Embarked’ and ‘Age’
c. Combine SubSp (siblings and spouse) and Parch (parents and children) variables to see if the number of dependents on a person gets affected by the outcome
d. Checked the correlation among variables:




PassengerId
Pclass
Age
SibSp
Parch
Fare
0.027
1
0.49
0.0011
0.019
0.58
0.034
0.49
1
0.092
0.061
0.34
0.0038
0.0011
0.092
1
0.31
0.17
0.043
0.019
0.061
0.31
1
0.23
0.58
0.34
1.0
0.8
0.6
0.4
0.2


Survival of passengers
Women
Men
Number of People
600
500
400
300
200
100
0
No
Yes
Survival


Pclass
1
2
3
1.0
0.8
0.6
0.4
0.2
0.0
24.0])
31.0])
80.0])
24.0])
31.0])
80.0])



e. Encoding categorical values of ‘Emarked’, ‘Sex’, ‘Title’ of into numerical data using map() [Unlike LabelCoding of tutorial]
f. Filling missing values of ‘Age’ column with the column that has a correlation on age. Finally replacing the value on that columns by median and considering columns 'Sex', 'Pclass','Title’ [Unlike tutorial where missing values were replaced by mean values based on Title column only]
g. Missing value handling of ‘Fare’ using median median value [Unlike tutorial]

Observation:

- I have created two separate files, one following the tutorial and another one with my changes.

- Most of the preprocessing part is same and can not be modified much. So I have kept it as it is following the tutorial
- I have applied SVM and Logistic regression to both the preprocessed data. For SVM, accuracy is almost the same between tutorial and my one. But for Logistic regression, my changed preprocessed data has given better accuracy compared to tutorial.

(b) [10 points]

A few students forgot to write their identification information such as name or student id on their final exams. What are some of the data preprocessing strategies that you will use to deal with these missing values?

Answer:

Before handling the missing data it is important to understand the nature or pattern of missing data to determine the treatment of missing data. So before finding the solution to handle missing identification information of students, we have to look deep down into the reason for data missing. Data can be missing in the following ways:

1. Missing Completely at Random: When there is no pattern in missing values and missing values are randomly distributed. So analyzing the missing identification values of students, we will look search whether there is any pattern. For example, if we can find that all the students with missing values are from a specific course or, their exam room is the same, then the missing data is not random. But if we don't find any pattern, then we can assume the data to be completely missing at random.
2. Missing At Random: Assume that value is missing for a subset sample of data. For instance, if identification values for students are missing for a specific subject only, then we can consider the missing type as random missing type.
3. Not Missing At Random: When the missing value has specific pattern or structure, we can not consider and treat the missing value as random. For student's data set, if ID is missing for all the students from a specific course, then missing values are cannot be treated as missing at random.

Now that we are done with the analysis of the missing value, we can move forward towards the solution to apply some preprocessing strategies to handle missing identification information. Several solutions can be applicable based on data.

1. When name/id any one information is missing, then we can assign a global constant e.g. “Unknown” to the missing values and can treat them as a new class.
2. If we found that small portion of data is missing, say only missing ID/Name for 20 students, then filling up the missing values manually will be feasible and won’t be a tedious task to perform.
3. We can choose the imputing missing value randomly method. Before that data can be divided data into sub category based on course or section or subject. Considering the category type, upper and lower limit of random function can be decided. In this way we can ensure better performance in pre processing step. As ID is missing, that we have to make sure that randomly selected values are unique. Name can be duplicated if ID information is not missing.
4. If only Name is missing in the student data set, then we can use student ID as the name also because ID is unique. Another alternative way is combining alphabet with student ID to assign as Name.
5. If we find that all the ID information is present in the data and name is not required for further processing on this data set, then we can ignore the missing Names.
6. If Name is present, but ID is missing, there is a problem. Because two students can have the same name but not same ID. In that case we can not ignore the missing values and can randomly assign unique IDs.
7. Using the entire or category wise data set a predictive model can be built to predict the missing identification values of students.
8. Some automated tools can be used. It can be fed with the entire student database and comparing with that data tool can automatically identify and replace the missing values.

3. (a) [10 points]

Install Hadoop on your system. Follow the simple map-reduce tutorial at https://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/

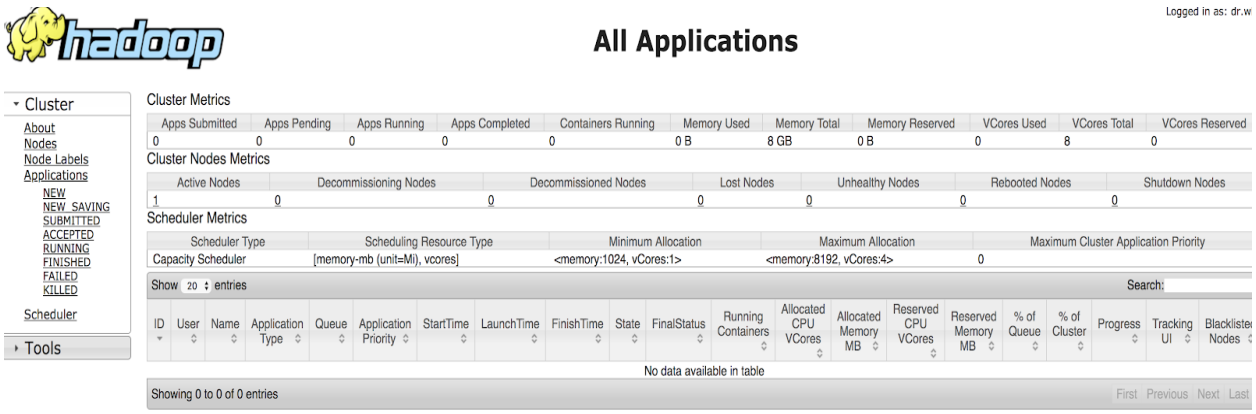
. Now write similar programs for this dataset https://www.kaggle.com/noriuk/us-education-datasets-unification-project#states_all_extended.csv

to determine two values: (i) the number of rows where the enrollment in grade 9 to 12 (Column name: GRADES_9_12_G) is less than 5,000 and (ii) between 10,000 and 20,000

Answer: Here is the screenshot of the Hadoop installation:

```
Anindyas-MacBook-Pro:hadoop-3.1.3 anindya$
Anindyas-MacBook-Pro:hadoop-3.1.3 anindya$
Anindyas-MacBook-Pro:hadoop-3.1.3 anindya$
Anindyas-MacBook-Pro:hadoop-3.1.3 anindya$
Anindyas-MacBook-Pro:hadoop-3.1.3 anindya$
Anindyas-MacBook-Pro:hadoop-3.1.3 anindya$ sbin/start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [Anindyas-MacBook-Pro.local]
2019-10-31 00:01:51,221 WARN util.NativeCodeLoader: Unable to load native-hadoop library
Anindyas-MacBook-Pro:hadoop-3.1.3 anindya$
Anindyas-MacBook-Pro:hadoop-3.1.3 anindya$
Anindyas-MacBook-Pro:hadoop-3.1.3 anindya$ jps
58304 DataNode
58199 NameNode
58442 SecondaryNameNode
58527 Jps
Anindyas-MacBook-Pro:hadoop-3.1.3 anindya$
Anindyas-MacBook-Pro:hadoop-3.1.3 anindya$
Anindyas-MacBook-Pro:hadoop-3.1.3 anindya$
Anindyas-MacBook-Pro:hadoop-3.1.3 anindya$ bin/hdfs dfs -mkdir /user
2019-10-31 00:02:21,522 WARN util.NativeCodeLoader: Unable to load native-hadoop library
Anindyas-MacBook-Pro:hadoop-3.1.3 anindya$ bin/hdfs dfs -mkdir /user/anindya
2019-10-31 00:02:35,936 WARN util.NativeCodeLoader: Unable to load native-hadoop library
Anindyas-MacBook-Pro:hadoop-3.1.3 anindya$
Anindyas-MacBook-Pro:hadoop-3.1.3 anindya$
Anindyas-MacBook-Pro:hadoop-3.1.3 anindya$ sbin/start-yarn.sh
Starting resourcemanager
Starting nodemanagers
Anindyas-MacBook-Pro:hadoop-3.1.3 anindya$
Anindyas-MacBook-Pro:hadoop-3.1.3 anindya$ jps
58304 DataNode
58930 NodeManager
58199 NameNode
58442 SecondaryNameNode
58830 ResourceManager
```

Here is the screenshot of resource-manager:



I have written the programs mapper.py, reducer_5000.py and reducer_ge_10k_le_20k.py for counting the number of rows where the enrollment in grade 9 to 12 (Column name: GRADES_9_12_G) is less than 5,000 and between 10,000 and 20,000.

File 1: mapper.py

File 2: reducer_5000.py

File 3: reducer_ge_10k_le_20k.py

Output:

(i) The number of rows where enrollment is less than 5000 is 53.

(ii) The number of rows where enrollment is between 10,000 and 20,000 is 52.

(b) [10 points]

Write the map programs and their corresponding reduce programs, both in python for some of the common data preprocessing tasks. No need to run them.

Answer:

Some major data preprocessing techniques are Data cleaning, data reduction and dimensionality reduction. I have chosen the State dataset from 3a to implement some of these preprocessing techniques. State data set has some redundant columns. So data reduction and dimensionality reduction techniques can be applied here. Following are my data preprocessing techniques on this data:

1. Aggregation of data by state: We can use the mapper in such a way that we output (state, columns for each year) and then use the reducer to compute sum or average of the values in each column. Thus we will get the sum or average of each column over both years for each state.

   Screenshot of mapper_aggregate_by_state.py:

```
#!/usr/bin/env python
"""mapper.py"""

import sys

# input comes from STDIN (standard input)
for line in sys.stdin:

    line = line.strip()
    # split the line into words
    words = line.split(',')
    if words[0] == 'PRIMARY_KEY':
        continue
    all_values = ''
    for i in range(3, len(words)):
        all_values += words[i]
        all_values += ','
    print '%s\t%s' % (words[1], all_values) # words[1] contains the name of the state
```

   Screenshot of reducer_aggregate_by_state.py:

```
#!/usr/bin/env python
"""reducer.py"""

from operator import itemgetter
import sys

prev_state = ""
prev_values = []

# input comes from STDIN
for line in sys.stdin:

    state, words = line.split('\t')

    values = words.split(',')

    # convert value (currently a string) to float
    if (state == prev_state):
        length = len(prev_values)
        for i in range(length):
            try:
                prev_values[i] += float(values[i])
            except ValueError:
                prev_values[i] = 0
    else:
        if prev_state != "":
            print prev_state
            for value in prev_values:
                print ',%s' (value)
        prev_state = state
        for value in values:
            try:
                prev_values.append(float(value))
            except ValueError:
                prev_values.append(0)
```

2. Aggregation of data by year: We can use the mapper to output the tuples (year, columns for each year) and then use the reducer to compute sum or average of the values for each column. Thus we will get the yearly sum or average in each column for all states.

   Screenshot of mapper_aggregate_by_year.py:

```
#!/usr/bin/env python
"""mapper.py"""

import sys

# input comes from STDIN (standard input)
for line in sys.stdin:

    line = line.strip()
    # split the line into words
    words = line.split(',')
    if words[0] == 'PRIMARY_KEY':
        continue
    all_values = ''
    for i in range(3, len(words)):
        all_values += words[i]
        all_values += ','
    print '%s\t%s' % (words[2], all_values) # words[2] contains the year
```

   Screenshot of reducer_aggregate_by_year.py:

```
#!/usr/bin/env python
"""reducer.py"""

from operator import itemgetter
import sys

prev_year = ""
prev_values = []

# input comes from STDIN
for line in sys.stdin:

    year, words = line.split('\t')

    values = words.split(',')

    # convert value (currently a string) to float
    if (year == prev_year):
        length = len(prev_values)
        for i in range(length):
            try:
                prev_values[i] += float(values[i])
            except ValueError:
                prev_values[i] = 0
    else:
        if prev_year != "":
            print prev_year
            for value in prev_values:
                print ',%s' (value/50.0) # printing the average of 50 states in a year
        prev_year = year
        for value in values:
            try:
                prev_values.append(float(value))
            except ValueError:
                prev_values.append(0)
```

4. (a) [10 points]

Describe five real-world applications in which regression can be used. For each of these applications, describe the y-value and the corresponding feature vector X. Also discuss whether linear regression can be used in each case.

Answer:

Following are the five real-world applications where regression can be used:

1. Prediction of crop yield: Regression can be applied to predict crop yield
    - Y-Value: Yield
    - Feature Vector X: Rainfall, Temperature, Humidity, Soil etc.
    - Linear Regression: Yes, linear regression can be used to predict crop yield
2. Prediction of product sales: Regression can be used to predict sales of products based on the past buying behavior of student
    - Y-Value: Sales

- Feature Vector X: Gender, Occupassion, Item purchased, Number of items purchased, Cost, Previous purchase date etc.
- Linear Regression: Yes, linear regression can be applied here

3. Regression in predicting cricket match score: Regression is being used in different sport analysis, say cricket. Here using regression prediction can be made on match score
    - Y-Value: Score
    - Feature Vector X: Date, Weather, Venue, Batting_team, Bowling_team, Batsman, Bowler, Previous_Score, Wickets, Overs, Striker, Non_Striker, Run_last_5_over, Wickets_last_5_over etc.
    - Linear Regression: Yes, data can be modeled and predicted using linear regression
4. Regression to predict GDP growth: Economists and financial people use regression to predict GDP growth to measure the growth and economic situation of a country
    - Y-Value: GDP growth
    - Feature Vector X: Stand and poor living index, Utilities, Industrales, Financials, Energy, Material, Technology, Natural resource, Health care, Consumers etc.
    - Linear Regression: Yes, GDP growth can be predicted using linear regression
5. Regression to predict Oil price: To analyze the stability of crude oil, regression model is being used to predict the rise and fall in oil price in different areas
    - Y-Value: Oil Price
    - Feature Vector X: Area, Period, Main Transportation, Total fuel consumption, Total oil rigs etc.
    - Linear Regression: Yes, for oil price prediction, linear regression can be used.

(b) [10 points]

List and briefly explain all the different ways we discussed in all the classes so far, that can help in reducing the dimensions of data, directly or indirectly.

Following are dimension methods that can be applied for dimension directly or indirectly:

1. Principal Component Analysis (PCA):
    - ❏ PCA is also known as Karhunen-Loeve, or K-L method.

- ❏ It is directly used for dimensionality reduction by projecting the original data onto a much smaller dimensional space.
- ❏ PCA is a statistical method that calculates principal component by orthogonally transforming a set of probable correlated variables to a set of linearly uncorrelated variables.
- ❏ Steps and explanations of PCA:
    - First calculate the covariance matrix C for the data points. Covariance measures from corresponding data set that how much the elements are correlated and moving in the same direction
    - Calculate the eigenvector and eigenvalue for matrix C.
    - The relation will be like this:

      [Covariance matrix].[Eigenvector]=[Eigenvalue].[Eigenvector]

    - Sort the eigenvectors in descending order according to their eigenvalues.
    - The eigenvector calculated from the largest eigenvalue is called the principal component 1. PC1 indicates the greatest variation direction from origin. Subsequents PC's are orthogonal to PC1.
    - Say the original data has n dimension. Then choose first k [k <= n] eigenvectors and that will be the new k dimensions of new data.
    - Final output implies that PCA has transformed n dimensional data points into k dimensional data.

2. Wavelet transformation:
    - ❏ Wavelet transformation is mostly a data compression technique used for image compression through dimensionality reduction.
    - ❏ It decomposes a signal and output signals are in different frequency level
    - ❏ Wavelet transformation makes nature clusters more distinguishable.
    - ❏ Two types of wavelet transformation: Discrete wavelet transformation (DWT) and Continuous wavelet transformation (CWT)
    - ❏ Discrete wavelet transformation (DWT):
        - The general procedure for applying a discrete wavelet transform uses a hierarchical pyramid algorithm.
        - Input vector length L must be an integer power of 2. Other wise padding is used to make it so.

- It halves the data at each iteration to increase the computational speed.
- Wavelet Coefficient is calculated at each step of the transformed data.

❏ Continuous wavelet transformation (CWT):

- First select a wavelet and compare it to a subsection from the starting of original signal.
- Calculate number C which represents how closely correlated the wavelet is with the subsection of the original signal. The higher value of C indicates higher similarity. C may be interpreted as a correlation coefficient. Results will depend on the chosen shape of wavelet.
- Shift the wavelet to the right and repeat the first two steps until the whole signal is covered.
- Scale the wavelet and repeat the first three steps.
- Repeat all the previous steps for all scales.

3. Attribute subset selection:
   ❏ Attribute subset selection is another effective method for dimensionality reduction
   ❏ It reduces the data set size by removing irrelevant or redundant attributes or dimensions.
   ❏ In data sometimes we have duplicated or similar type of attribute containing the same information, these are called redundant attributes.
   ❏ Sometimes data has attributes that doesn't contain any useful information and we remove those irrelevant attributes from further processing.
   ❏ Heuristic techniques for attribute subset selection:
      - Stepwise Forward selection: This procedure starts with an empty data set Then iteratively identifies important attributes and keep adding to the set. Final output is a reduced dimension set.
      - Stepwise Backward elimination: The procedure starts with the full set of attributes and it removes the irrelevant worst attributes in the subsequent steps from remaining in the set.
      - Combination of forward selection and backward elimination: This stepwise method is a combination of stepwise forward selection stepwise backward elimination method. At each step this technique eliminates the worst attributes and selects the best attributes.

- Optimal branch and bound: It is a combination of attribute elimination and backtracking method.

4. Decision tree:
   - ❏ Decision tree is the simplest algorithm with no parameter .
   - ❏ Different types of decision trees are: CART, ID3, C4.5
   - ❏ Tree leaves represents the classification and branches represent test on feature to reach to the classification.
   - ❏ Decision tree is used for feature selection which directly or indirectly leads to dimensionality reduction.
   - ❏ Information gain , Gini index etc. are used to select the next attribute for node split.
   - ❏ To avoid overfitting sometimes we don't grow the tree after some point when data doesn't yield an improvement
   - ❏ Sometimes after growing the full tree we prune the unnecessary nodes
   - ❏ Both the cases reduce dimension of data.

5. Log linear model:
   - ❏ Log linear model is basically used for numerosity reduction. But due to its nature can be applied to achieve dimensionality reduction also.
   - ❏ This model represents a function whose logarithm is in linear relation with the with the combination of parameters
   - ❏ Representation function is: $Log(\mu i) = \alpha + \beta 1 \ * X1i... + \beta n \ * Xni$
   - ❏ Log linear models calculates the probability of each point in a multidimensional space for a set of discretized attributes and which is based on a comparatively smaller subset of dimensional combination.
   - ❏ So we can construct the higher dimension data space from lower dimensional space.
   - ❏ That's why log linear model can be used for dimensionality reduction method

6. Data Transformation Methods:
   - ❏ In data transformation, data are transformed into a form to map to a new set of replacement values
   - ❏ New transformation maintain the property that old values can be identified with one of the new values
   - ❏ Data transformation techniques are not directly used for dimensionality reduction. It itself does not reduce data but help in data reduction.

- ❏ It can be considered as a steps for reduction, e.g. fourier transformation
- ❏ So data transformation method indirectly helps in dimensionality reduction.

5. (a) [10 points]

Below is an excellent blog on three data visualization techniques.

https://towardsdatascience.com/3-awesome-visualization-techniques-for-every-dataset-9737eecacbe8

Use the techniques on one or more datasets available on Kaggle.

Answer:

File: hw3_TwoKaggleDataSet_Visualization.correlation_pairplot_swarm

Dataset1: adult_census_income.csv

Dataset2: IRIS.csv

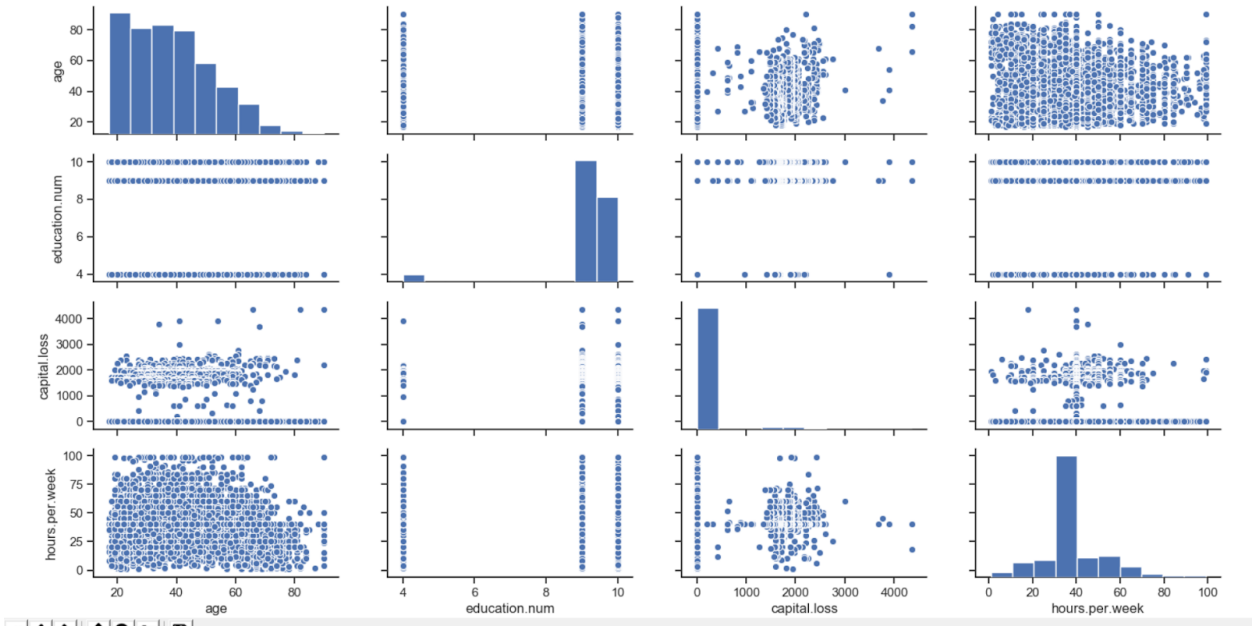
Output: I have applied the correlation, pairplot and swarm visualization techniques on two kaggle data set named: adult_census_income amd iris data sets. Plots are below:

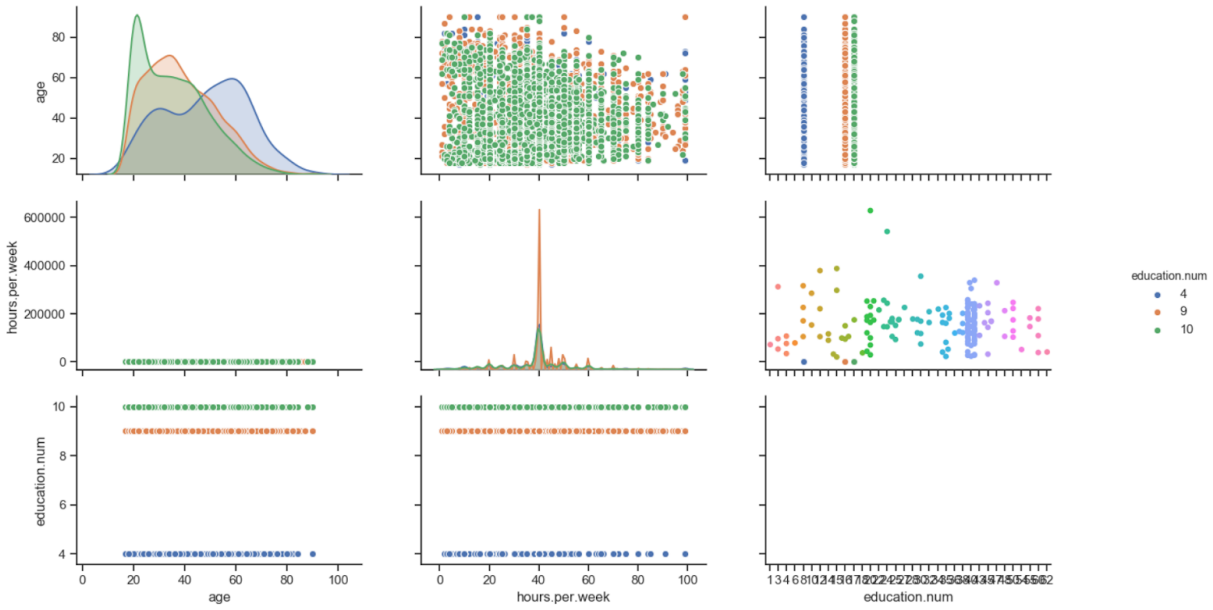
1.

correlation:

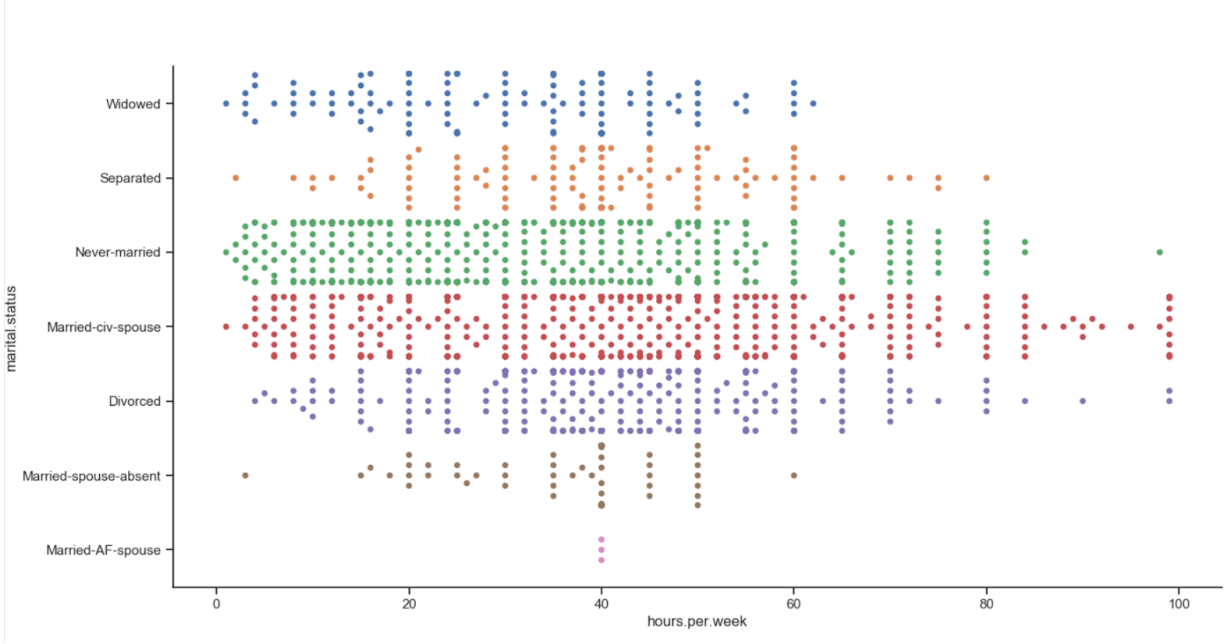

age
fnlwgt
education.num
capital.gain
capital.loss
hours.per.week
1.00 -0.08 0.04 0.08 0.06 0.07
-0.08 1.00 -0.04 0.00 -0.01 -0.02
0.04 -0.04 1.00 0.12 0.08 0.15
0.08 0.00 0.12 1.00 -0.03 0.08
0.06 -0.01 0.08 -0.03 1.00 0.05
0.07 -0.02 0.15 0.08 0.05 1.00
age
fnlwgt
ation.num
apital.gain
apital.loss
.per.week
0.8
0.6
0.4
0.2
0.0

pairplot :


age
education.num
capital.loss
hours.per.week

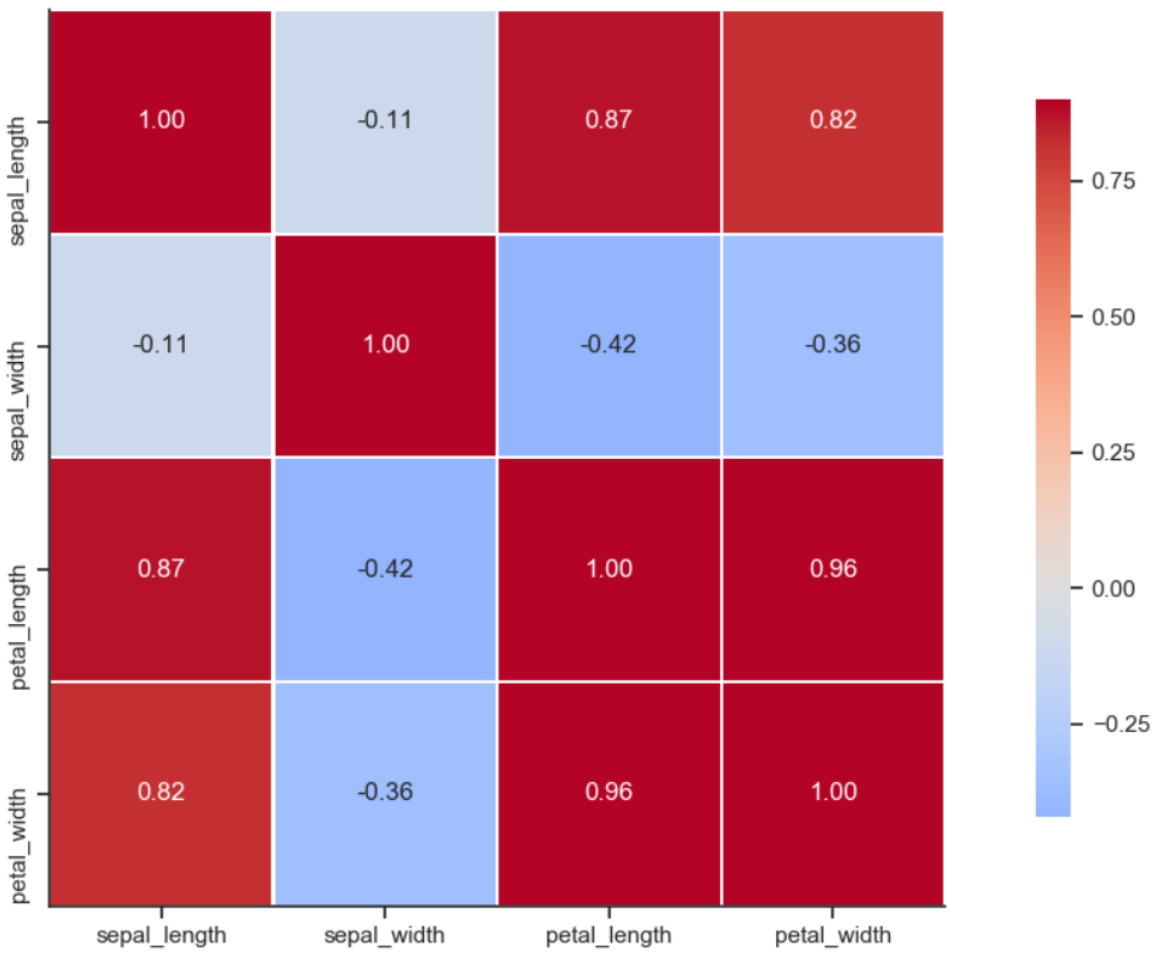

age
hours.per.week
education.num
education.num
4
9
10

swarm :

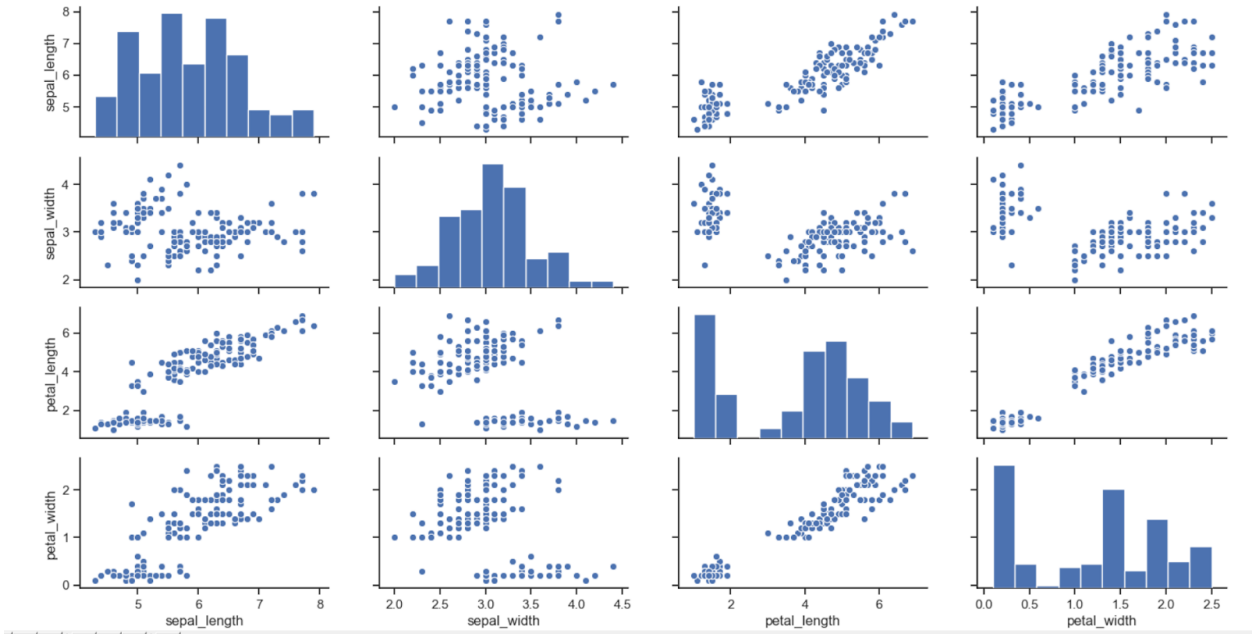


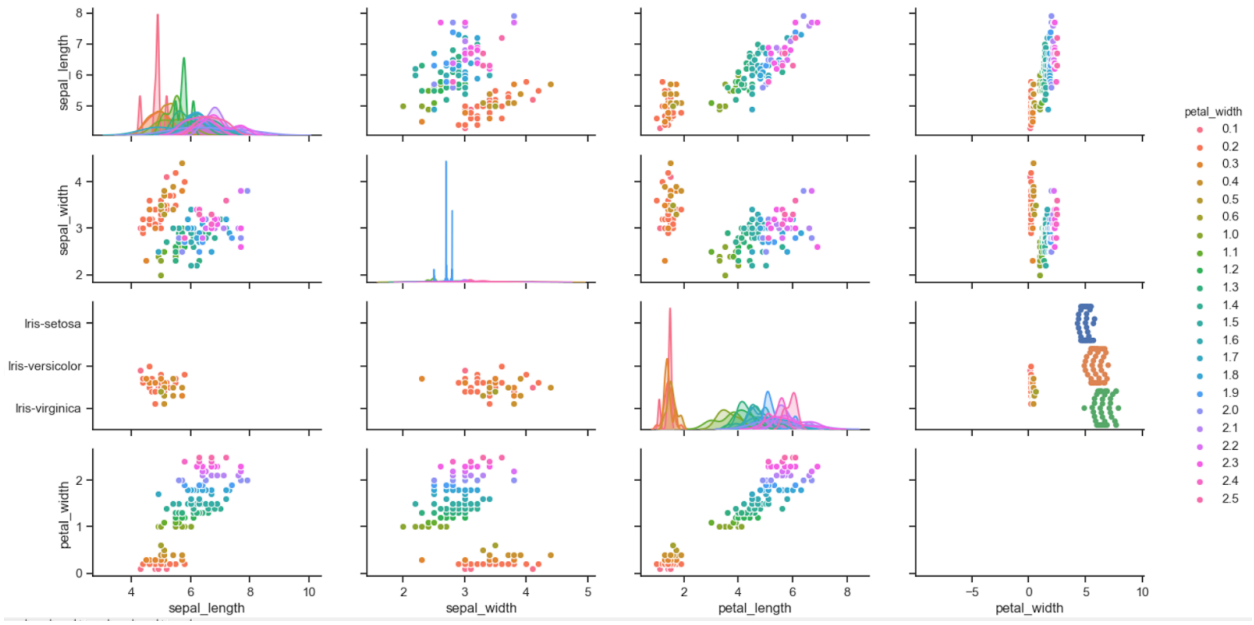
2.

correlation:

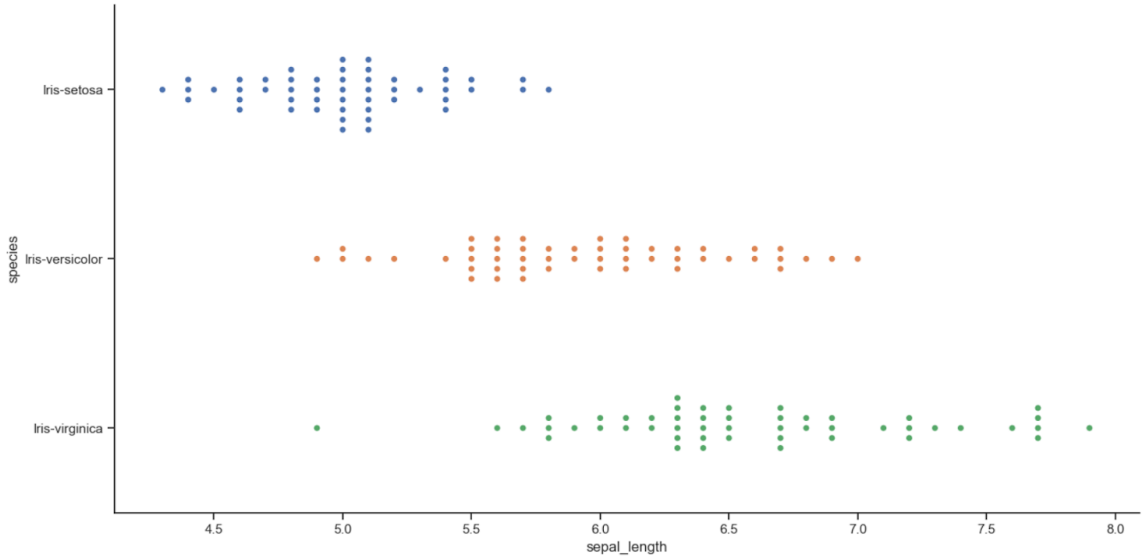

1.00
-0.11
0.87
0.82
-0.11
1.00
-0.42
-0.36
0.87
-0.42
1.00
0.96
0.82
-0.36
0.96
1.00
sepal_length
sepal_width
petal_length
petal_width
sepal_length
sepal_width
petal_length
petal_width
0.75
0.50
0.25
0.00
−0.25

pairplot :


sepal_length
sepal_width
petal_length
petal_width


sepal_length
sepal_width
petal_length
petal_width
Iris-setosa
Iris-versicolor
Iris-virginica
petal_width

swarm :



(b) [10 points]

Assume there's a data set that has just three columns (two features and one label) and four rows (items). The four vectors corresponding to the items are at the corners of a square. The two vectors at the ends of one diagonal of this square belong to one class and the other two vectors on the other diagonal belong to the second class. Is this data separable by a straight line? Which algorithm that you studied in the class would you choose if you were to come up with a classifier for this toy data set and why?

Answer:

The question describes that the data set has two features and one label and four rows. Moreover the four vectors corresponds to the four corners of a square. Considering all these, without loss of generality, let's assume that the following table is our data set.

Row No.	X1	X2	Y
a1	0	0	

a2	1	0	
a3	1	1	
a4	0	1	

Here X1, X2 are two features and Y is the class (not decided yet). It's easy to under that a1(0,0), a2 (1,0), a3 (1,1), a4 (0,1) are four corners of a square. Now according to problem, vector of a diagonal belongs to the same class. Based on that, I am assigning the class below.

Row No.	X1	X2	Y
a1	0	0	C1
a2	1	0	C2
a3	1	1	C1
a4	0	1	C2

So in the above table, two corners of the same diagonal a1(0,0)-a3 (1,1) have been assigned to class C1 and the remaining corners are assigned to class C2.

From my understanding, this data is not separable by a straight line in this state. But we can apply SVM here to make the data linearly separable by applying a kernel a trick. After applying the kernel function on the above data set, we can separate it by a straight line.

For the above data set, I shall apply the following kernel function to transform the vectors in a way so that they are linearly separable by a straight line.

$$K(X1,\ X2) = \{(X1\ *\ (1 - X1)), (X2\ *\ (1 - X2))\}$$

After applying kernel function on the above dataset, we get :

Row No.	X1	X2	K-X1	K-X2	Y
a1	0	0	0	0	C1

a2	1	0	0	1	C2
a3	1	1	0	0	C1
a4	0	1	1	0	C2

So final outcome of kernel trick function on feature vectors is : a1(0,0) ⇒ a1(0,0), a2 (1,0) ⇒ (0,1), a3 (1,1)⇒ a3 (0,0) and a4 (0,1) ⇒ a4 (1,0). Here we can see that, for class C1, a1 remains same but a3 changes its position comes to the side of a1. For class C2, both the points just interchanges their position. Now from the new orientation of points it is clear that they are linearly separable as C1 class points resides on the same side and C2 class points just interchanged their position.

From the above hypothesis of the scenario, this data set is linearly separable by a straight line using SVM with a kernel trick. For other data sets also with same condition of forming a square we can do that only by appropriately developing the correct kernel function and applying it.