DATA MINING CSC 6740
HOMEWORK 1

MOHAMMED SHOEBUDDIN HABEEB
002521737

Ans 1:
Data science is a field where data in its raw form is processed into information. Domain knowledge is the knowledge about the environment in which the data is processed to reveal secrets of the data. The true power of an algorithm and data can be harnessed when we have some form of domain knowledge. Needless to say, the accuracy of the model also increases with the use of such knowledge of data. One major application of such domain knowledge is prominent in feature engineering. Feature engineering is creating features using the domain knowledge to optimize the machine learning algorithms.

For example, the knowledge of the automobile industry when working with the relevant data can be used Let's say we have two features Horsepower and RPM from which we can create an additional feature like Torque from the formula

$$TORQUE = HP \times 5252 \div RPM$$

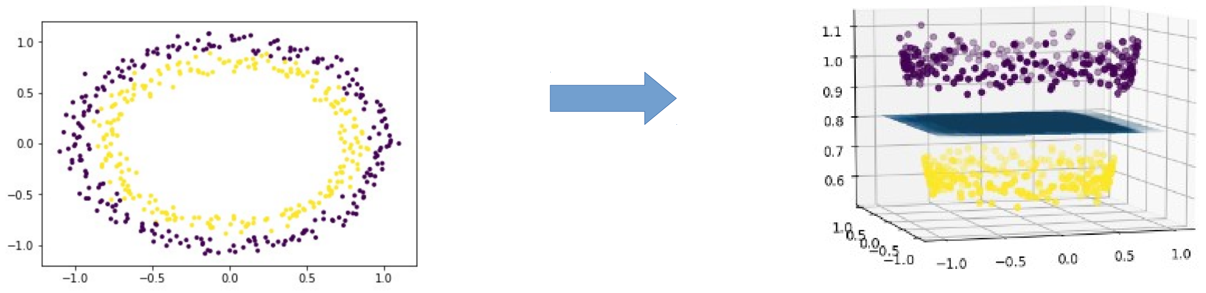
This additional feature can help add more insights into the machine learning model.

Ans 2:
Data science techniques try to extract information from chunks of data. These databases can get incredibly massive and usually contain data of all sorts, from comments left on social media to numbers coming from analytic dashboards. That vast amount of information is heterogeneous by nature, which means that they don't share the same structure. Raw data can have missing or inconsistent values as well as present a lot of redundant information. Data is said to be unclean if it is missing attribute, attribute values, contain noise or outliers and duplicate or wrong data. Presence of any of these mistakes, redundancies, missing values, and inconsistencies all compromise the integrity of the set. Thus, before using that data for the purpose you want, you need it to be as organized and "clean" as possible.
Major approaches to preprocess data are Data Cleaning, Data Transformation and Data Reduction.
One such method is cleaning the data by finding missing values. This could be a very sensitive topic as usually people tend to remove data records which have missing values. This could lead to a potential data loss and consequently a very biased classifier is built on the distorted data. To tackle this issue missing data can be imputed with mean or median of the the attribute. Another major issue is presence of random scaled data. For example if we compare two attributes with very different scale of measurements or properties the classifier and the deductions we make can be very skewed. Normalization of the dataset is really important to tackle the issue.
Another Data selection methodology could be when sampling a subset from the huge data. We have to make sure that we sample a representative subset. Randomly sampling is a common technique but it is error prone as there is as high a chance to have unrepresentative data. For this issue picking a stratified sample with classes representative of the attributes is a good step forward. Imbalance of classes is also a major problem as the majority classes tend to make the classifier biased and skewed. Undersampling and oversampling techniques are some of the ways to restore balance in the dataset.

Ans 3)
Within the field of machine learning, there are two main types of tasks: supervised, and unsupervised. The main difference between the two types is that supervised learning is done using a ground truth, or in other words, we have prior knowledge of what the output values for our samples should be. Therefore, the goal of supervised learning is to learn a function that, given a sample of data and desired outputs, best approximates the relationship between input and output observable in the data. Unsupervised learning, on the other hand, does not have labeled outputs, so its goal is to infer the natural structure present within a set of data points.
The major techniques in supervised learning are classification and regression. We can use tools like KNN, Naive Bayes, SVM, Decision trees and neural networks.

Ans 4) [1]
A learning model that summarizes data with a set of parameters of fixed size (independent of the number of training examples). No matter how much data you throw at a parametric model, it won't change its mind about how many parameters it needs. the complexity of the model is bounded even if the amount of data is unbounded. This makes them not very flexible
Nonparametric methods are good when you have a lot of data and no prior knowledge, and when you don't want to worry too much about choosing just the right features.
Linear models such as linear regression, logistic regression, and linear Support Vector Machines are typical examples of a parametric "learners;" here, we have a fixed size of parameters (the weight coefficient). In contrast, K-nearest neighbor, decision trees, or RBF kernel SVMs are considered as non-parametric learning algorithms since the number of parameters grows with the size of the training set. – K-nearest neighbor and decision trees, that makes sense, but why is an RBF kernel SVM non-parametric whereas a linear SVM is parametric? In the RBF kernel SVM, we construct the kernel matrix by computing the pair-wise distances between the training points, which makes it non-parametric.
To summarize, the trade-offs between parametric and non-parametric algorithms are in computational cost and accuracy.

Ans 5)
Distribution of data tells us about the linearity and innate properties of the dataset. If we can seperate the dataset classes easily by a straight line the data is called linearly seperable. If the dataset can be seperated by a non linear function then it is called a non linear model. We can perform regression on Non linear data by transformation and fitting the data to transformed axes. Consequently, nonlinear regression can fit an enormous variety of curves. However, because there are so many candidates, you may need to conduct some research to determine which functional form provides the best fit for your data. A random forest regression is considered a non-linear model. Random forest models are ensemble learning methods for regression which grow a forest of regression trees and then average the outcomes.
In the case of classification, building a classifier is simple in case of linear data but in non linear data SVM offers the flexibility to perform classification. SVM is a linear classifier which learns an (n – 1)-dimensional classifier for classification of data into two classes. However, it can be used for classifying a non-linear dataset. This can be done by projecting the dataset into a higher dimension in which it is linearly separable

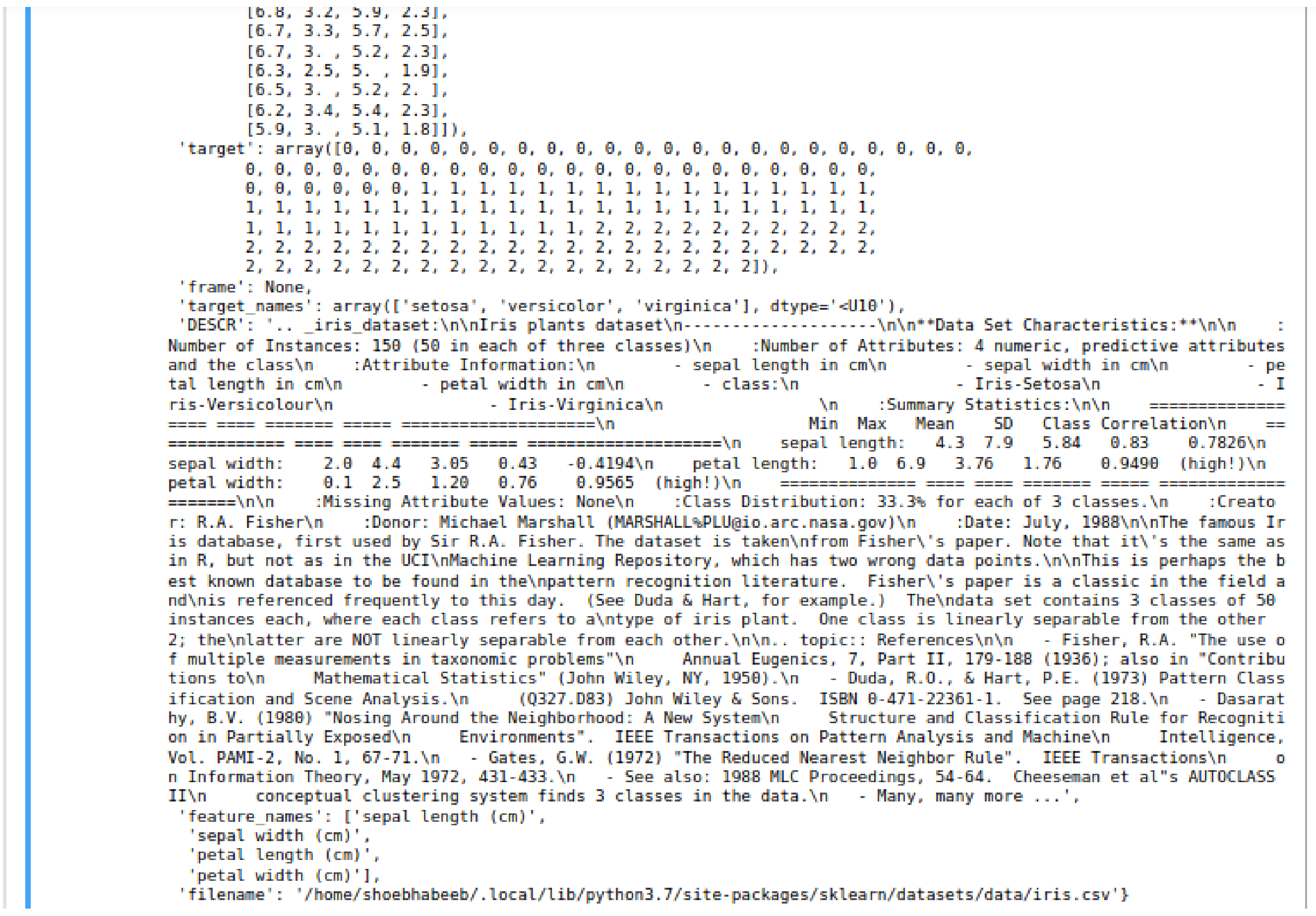


Linear models are often useful approximations to nonlinear relationships as long as we restrict our attention to realistic and relatively modest variations in the variables. All linear are not always useful mainly because they tend to combine attributes and think in terms of only linearly seperable form. This could lead to high errors and the whole point of our minimizing the error is defeated.
Linearly seperable datasets are easy to predict as they have the assumption that the hyperplane/line separating them follows a straight line equation no matter the amount of parameters. But for non linear this assumption doeasnt hold which leaves us with a challenge to find the relationships among the dependent variables. New ways of visualization would essentially would be transforming the data using a non linear function to get a resultant better performing classifier.

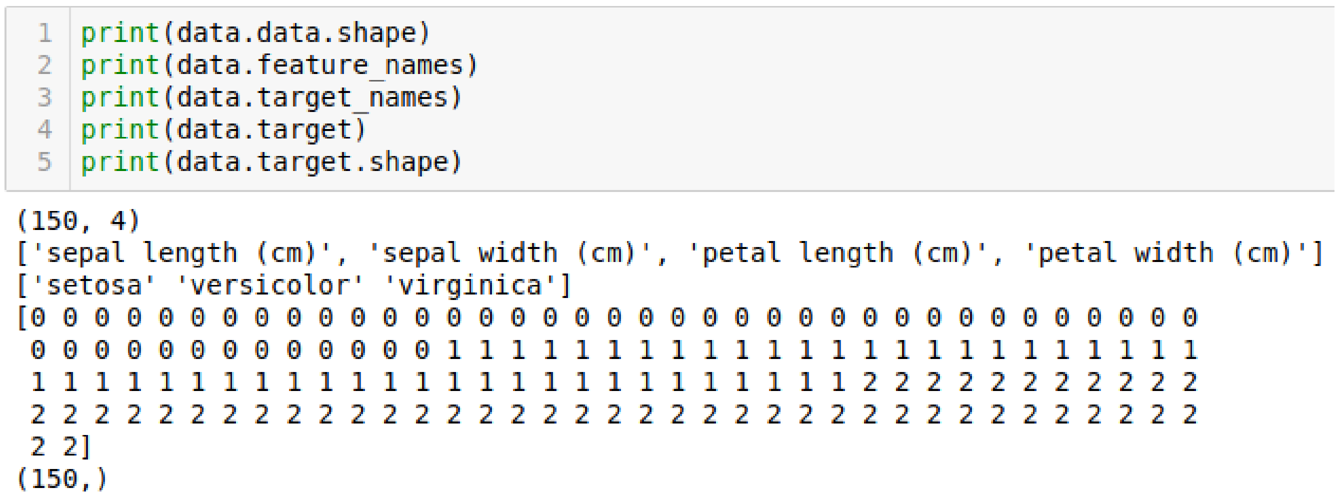
REFERENCES :
[1] http://mlss.tuebingen.mpg.de/2015/slides/ghahramani/gp-neural-nets15.pdf

PROGRAMMING

1) Data is a distionary



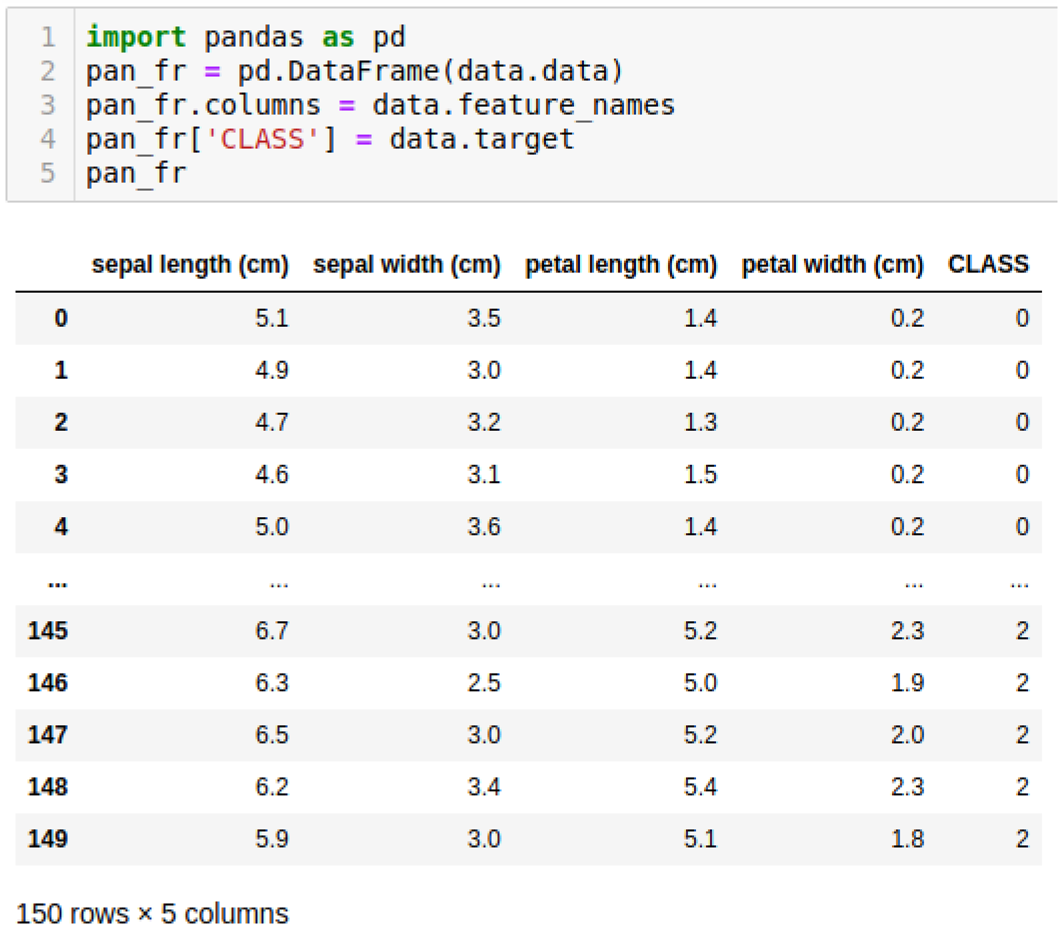
```
        [6.8, 3.2, 5.9, 2.3],
        [6.7, 3.3, 5.7, 2.5],
        [6.7, 3. , 5.2, 2.3],
        [6.3, 2.5, 5. , 1.9],
        [6.5, 3. , 5.2, 2. ],
        [6.2, 3.4, 5.4, 2.3],
        [5.9, 3. , 5.1, 1.8]]),
 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]),
 'frame': None,
 'target_names': array(['setosa', 'versicolor', 'virginica'], dtype='<U10'),
 'DESCR': '.. _iris_dataset:\n\nIris plants dataset\n--------------------\n\n**Data Set Characteristics:**\n\n    :
Number of Instances: 150 (50 in each of three classes)\n    :Number of Attributes: 4 numeric, predictive attributes
and the class\n    :Attribute Information:\n        - sepal length in cm\n        - sepal width in cm\n        - pe
tal length in cm\n        - petal width in cm\n        - class:\n                - Iris-Setosa\n                - I
ris-Versicolour\n                - Iris-Virginica\n                \n    :Summary Statistics:\n\n    ==============
==== ==== ======= ===== ====================\n                    Min  Max   Mean    SD   Class Correlation\n    ==
============ ==== ==== ======= ===== ====================\n    sepal length:   4.3  7.9   5.84   0.83    0.7826\n
sepal width:    2.0  4.4   3.05   0.43   -0.4194\n    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)\n
petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)\n    ============== ==== ==== ======= ===== ============
=======\n\n    :Missing Attribute Values: None\n    :Class Distribution: 33.3% for each of 3 classes.\n    :Creato
r: R.A. Fisher\n    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)\n    :Date: July, 1988\n\nThe famous Ir
is database, first used by Sir R.A. Fisher. The dataset is taken\nfrom Fisher\'s paper. Note that it\'s the same as
in R, but not as in the UCI\nMachine Learning Repository, which has two wrong data points.\n\nThis is perhaps the b
est known database to be found in the\npattern recognition literature.  Fisher\'s paper is a classic in the field a
nd\nis referenced frequently to this day.  (See Duda & Hart, for example.)  The\ndata set contains 3 classes of 50
instances each, where each class refers to a\ntype of iris plant.  One class is linearly separable from the other
2; the\nlatter are NOT linearly separable from each other.\n\n.. topic:: References\n\n   - Fisher, R.A. "The use o
f multiple measurements in taxonomic problems"\n     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contribu
tions to\n     Mathematical Statistics" (John Wiley, NY, 1950).\n   - Duda, R.O., & Hart, P.E. (1973) Pattern Class
ification and Scene Analysis.\n     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.\n   - Dasarat
hy, B.V. (1980) "Nosing Around the Neighborhood: A New System\n     Structure and Classification Rule for Recogniti
on in Partially Exposed\n     Environments".  IEEE Transactions on Pattern Analysis and Machine\n     Intelligence,
Vol. PAMI-2, No. 1, 67-71.\n   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions\n     o
n Information Theory, May 1972, 431-433.\n   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS
II\n     conceptual clustering system finds 3 classes in the data.\n   - Many, many more ...',
 'feature_names': ['sepal length (cm)',
  'sepal width (cm)',
  'petal length (cm)',
  'petal width (cm)'],
 'filename': '/home/shoebhabeeb/.local/lib/python3.7/site-packages/sklearn/datasets/data/iris.csv'}
```



```python
print(data.data.shape)
print(data.feature_names)
print(data.target_names)
print(data.target)
print(data.target.shape)
```

```
(150, 4)
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
['setosa' 'versicolor' 'virginica']
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
(150,)
```

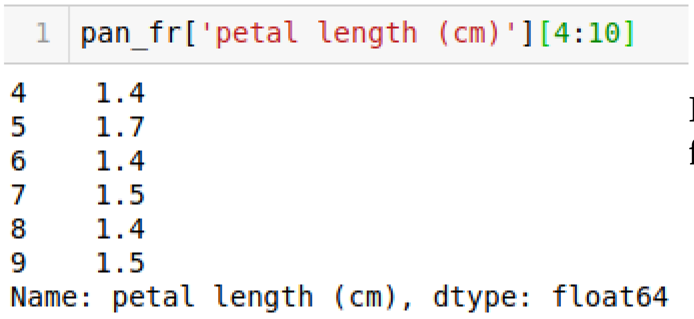
2)



```python
import pandas as pd
pan_fr = pd.DataFrame(data.data)
pan_fr.columns = data.feature_names
pan_fr['CLASS'] = data.target
pan_fr
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	CLASS
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

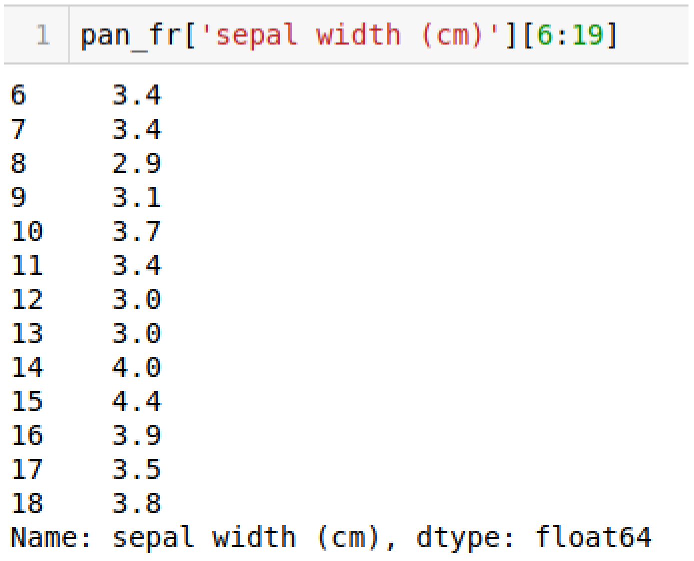
150 rows × 5 columns



```python
pan_fr['petal length (cm)'][4:10]
```

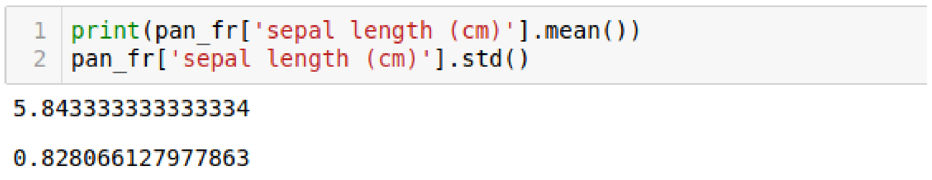
```
4    1.4
5    1.7
6    1.4
7    1.5
8    1.4
9    1.5
Name: petal length (cm), dtype: float64
```

Because the index in python starts from 0



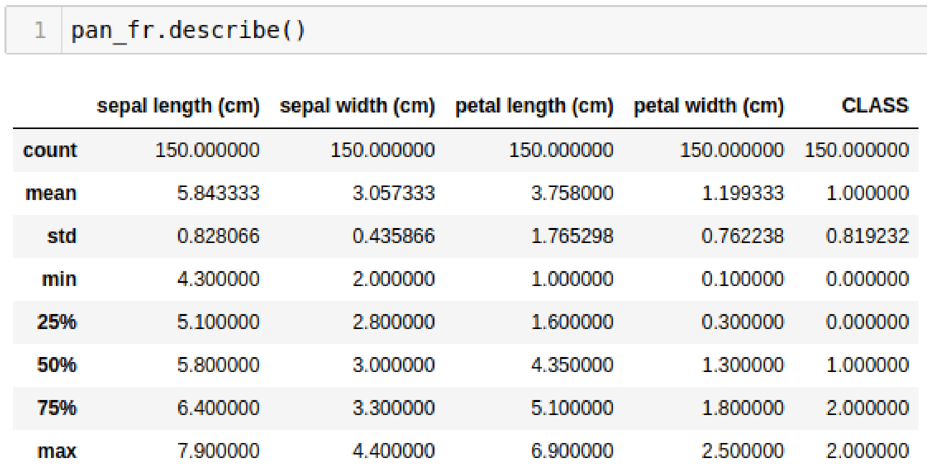
```python
pan_fr['sepal width (cm)'][6:19]
```

```
6     3.4
7     3.4
8     2.9
9     3.1
10    3.7
11    3.4
12    3.0
13    3.0
14    4.0
15    4.4
16    3.9
17    3.5
18    3.8
Name: sepal width (cm), dtype: float64
```



```python
print(pan_fr['sepal length (cm)'].mean())
pan_fr['sepal length (cm)'].std()
```

```
5.843333333333334

0.828066127977863
```



```python
pan_fr.describe()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	CLASS
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

3)

```
import pandas as pd
nba_df = pd.read_csv("nba.csv")
nba_df
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
...	...	...	...	...	...	...	...	...	...
453	Shelvin Mack	Utah Jazz	8.0	PG	26.0	6-3	203.0	Butler	2433333.0
454	Raul Neto	Utah Jazz	25.0	PG	24.0	6-1	179.0	NaN	900000.0
455	Tibor Pleiss	Utah Jazz	21.0	C	26.0	7-3	256.0	NaN	2900000.0
456	Jeff Withey	Utah Jazz	24.0	C	26.0	7-0	231.0	Kansas	947276.0
457	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

458 rows × 9 columns

```
nba_df.index[nba_df.isnull().all(1)
```

```
Int64Index([457], dtype='int64')
```

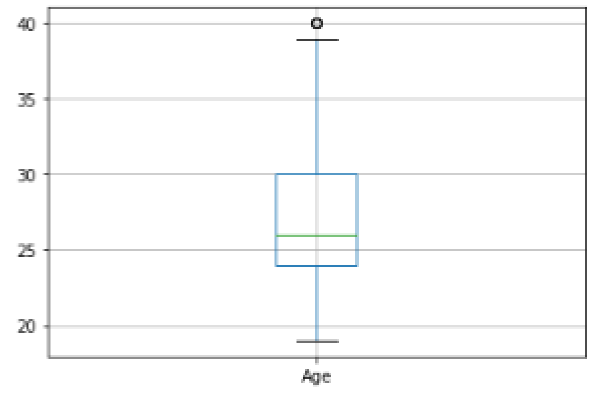
```
print(nba_df.shape)
print(nba_df['College'].isna().sum())
nba_df['Age'].isna().sum()
```

```
(458, 9)
85

1
```

missing values in College column: 85
missing values in Age column: 1

```
print(nba_df['Age'].describe())
nba_df.boxplot(column = ['Age'])
```

```
count    457.000000
mean      26.938731
std        4.404016
min       19.000000
25%       24.000000
50%       26.000000
75%       30.000000
max       40.000000
Name: Age, dtype: float64

<AxesSubplot:>
```



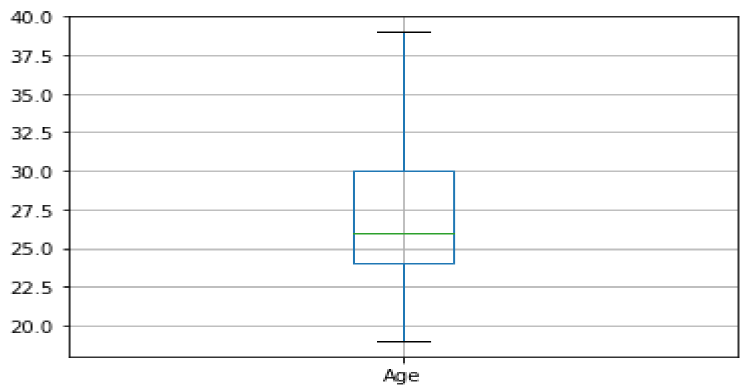
From the describe we see that the max and min in the Age columns are skewed and so we plot a box plot which shows the presence of outliers.
Also there is a NaN value in the Age column.

The max value should be 30 + 1.5*6 = 39 and min value should be 24-1.5*6 = 15

We can bin all the values to certain range by clamping them on both max and min sides. This can help remove the outliers. Another way is we can identify and totally remove those outliers. As for NaN values we can either remove the row or replace with mean of the column.

```
nba_df['Age'] = nba_df['Age'].clip(upper = 39)
print(nba_df['Age'].describe())
nba_df.boxplot(column = ['Age'])
```

```
count    457.000000
mean      26.932166
std        4.385207
min       19.000000
25%       24.000000
50%       26.000000
75%       30.000000
max       39.000000
Name: Age, dtype: float64

<AxesSubplot:>
```



There is not much change in the statistic as mean and std remain almost the same but the max has now shifted.

To remove the NaN values identify their index and remove them from the dataframe

```
print(nba_df['Age'])
nba_df['Age'] = nba_df['Age'][:-1]
print(nba_df['Age'][:-1])
```

```
0      25.0
1      25.0
2      27.0
3      22.0
4      29.0
       ...
453    26.0
454    24.0
455    26.0
456    26.0
457     NaN
Name: Age, Length: 458, dtype: float64
0      25.0
1      25.0
2      27.0
3      22.0
4      29.0
       ...
452    20.0
453    26.0
454    24.0
455    26.0
456    26.0
Name: Age, Length: 457, dtype: float64
```

Another method to handle missing values in by filling them with mean

```
print(nba_df['Age'].isnull().sum())
nba_df['Age'].fillna(nba_df['Age'].mean(), inplace = True)
nba_df['Age'].isnull().sum()
```

```
1

0
```

```
print(nba_df['Salary'].isnull().sum())
nba_df['Salary'].fillna(nba_df['Salary'].mean(), inplace = True)
nba_df['Salary'].isnull().sum()
```

```
12

0
```

There is a marginal change in statistic

```
nba_df['Age'].describe()
```

```
count    458.000000
mean      26.932166
std        4.380406
min       19.000000
25%       24.000000
50%       26.000000
75%       30.000000
max       39.000000
Name: Age, dtype: float64
```

The covariance matrix is a square and symmetric matrix that describes the covariance between two or more random variables. The diagonal entries of the covariance matrix are the variances and the other entries are the covariances. For this reason, the covariance matrix is sometimes called the variance-covariance matrix. An interesting use of the covariance matrix is in the Mahalanobis distance, which is used when measuring multivariate distances with covariance. It does that by calculating the uncorrelated distance between a point x
to a multivariate normal distribution with the following formula

$$DM(x)=\sqrt{(x-\mu)^T C^{-1}(x-\mu)}$$

where μ is the mean and C is the covariance of the multivariate normal distribution

```
covMatrix = np.cov(nba_df['Age'], nba_df['Salary'])
covMatrix
```

```
array([[1.91879588e+01, 4.78246234e+06],
       [4.78246234e+06, 2.66268972e+13]])
```

********** FIN **********