Pandas Essential for Data Science

Muhammad Affan Alim

Brief introduction of Pandas

- Pandas is a newer package built on top of the NumPy, and provides an efficient implementation of a DataFram
- DataFrames are essentially multidimensional arrays with attached row and column labels
- Sometime heterogeneous types and/or missing data
- The three fundamentals of pandas data structure are "Series", "DataFrame", and "Index"

Pandas Library and version

[ ]: import pandas as pd

[ ]: print(pd.__version__)

The pandas Series object

- A pandas "Series" is a one-dimensional array of indexed data.
- It can be created from a list or array as follows:

```
[ ]: data = pd.Series([0.25, 0.5, 0.75, 1.0])
[o/p]: 0     0.25
       1     0.5
       2     0.75
       3     1.00
      dtype: float64
```

The pandas Series object cont…

- We can access with the "values" and "index" attributes. The values are simply a familiar NumPy array

```
[ ]: data.values
[o/p]: array([0.25, 0.5, 0.75, 1.0])

[ ]: data.index
[o/p]: RangeIndex(start=0, stop=4, step=1)
```

Feature Engineering

Encoding Technique

Encoding Categorical data

- Dataset contains some categorical data in qualitative nature
- It is easily understandable by human but encode these into numeric values for machine learning model
- Example
- Color, place of birth, fruit etc.

Index	Country
1	Pak
2	US
3	Pak
4	UK
5	Us
6	Singapore
7	UK

Encoding Categorical data

- these categories are unordered and when do the order then it penalize the model like pak= 1, US = 2 etc.
- can't assign any arbitrary number to any value for machine learning model

Index	Country	encoding
1	Pak	1
2	US	2
3	Pak	1
4	UK	3
5	US	2
6	Singapore	4
7	UK	3

Encoding Categorical data

- values can encode the additional binary features corresponding the each value respect or not
- Example

Index	Country
1	' Pak '
2	'USA'
3	'UK'
4	'UK'
5	'France'
...	...

Index	C_Pak	C_USA	C_UK	C_France
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	1	0
5	0	0	0	1
...	...	...	...	...

Encoding Categorical data

- In doing so your model can leverage the information that what country is given without inference any order between the different options
- There are two main methods for encoding techniques
  1. One-Hot encoding: *n*-category with *n*-feature
  2. Dummy encoding: *n*-category with (*n*-1)-feature

Encoding Categorical data

- Both models create huge number of columns being created
- Example:

- <u>One-hot encoding</u>

dfNew1 = pd.get_dummies(dfNew,columns=['Embarked'], prefix='E')

- <u>Dummy encoding</u>

dfNew2 = pd.get_dummies(dfNew,columns=['Embarked'], drop_first = True, prefix = 'E')

Encoding Categorical data

- **Limit your columns**
- What values to included? First creating the mask of the values which is less than n times in values.

[]: Counts = dfNew['Embarked'].value_counts()

Encoding Categorical data

- **First create the mask**

```
[]: mask = dfNew['Embarked'].isin(counts[counts<170].index)
[]: dfNew['Embarked'][mask] = 'other'
[]: print(pd.value_count(dfNew['Embarked']))
```

Feature Engineering

Dealing with Numeric Variables

Numeric Variables

- If data has all numeric values, but it also allows to improve the features
- Numeric features, such as
  age, price, counts, geospatial data
- It shows different characteristics, few consideration and possible feature engineering can be improved it

Numeric Variables cont…

- **Example**
- Check either the magnitude is the most important or just direction
- Data set of the health and safety rating that contains the number of times the restaurant major violations.

Numeric Variables cont…

- Consider the example

Index	Resturant_Id	Number of violation
0	RS_1	0
1	RS_2	0
2	RS_3	2
3	RS_4	1
4	RS_5	0
5	RS_6	4
6	RS_7	4
7	RS_8	1
8	RS_9	0
9	RS_10	2

Numeric Variables cont…

- **First method**
- Create a binary column that a restaurant committed the violation or not

[]: df[‘Binary_violation’] = 0

[]: df.loc[df[‘Number_of_violations’]>0, ‘Binary_violation’]=1

Numeric Variables cont…

Second Method-Binning Numeric Variable

- This is often useful such as age, weight, etc.

```
[]: import numpy as pd
[]: df['Binned_Grouped'] = pd.cut(df['Number_of_violations'],
                                  bins=[-np.inf,0,2,np.inf],
                                  labels=[1,2,3])
```

Feature Engineering
Missing Data

Dealing with Missing Values

- First recognize why the missing values occurs ?
- if it is confirmed that missing values occurs at random not being intentially be omitted.
- Most statistic and sound approach is called “complete case analysis” or “list wise deletion”
- In this method the record fully excluded from the model.

Dealing with Missing Values cont…

-

Index	Survey date	Converted salary	Hobby
1	2/28/18	NaN	Yes
2	6/28/18	7084.0	Yes
3	6/6/18	NaN	No
4	5/9/18	21426.0	Yes
5	4/12/18	41671.0	Yes

Dealing with Missing Values cont…

- List wise deletion in python

```
# Drop all rows with atleast one missing values in any column
[]: df.dropna(how='any')

# Drop in specific column
[]: df.dropna(subset=['converted_salary'])
```

Dealing with Missing Values cont…

- Issue with list wise deletion
- Several drawbacks are exist of list wise deletion

1. It deletes a valid data points
2. Relies on randomness: If missing values not omitted randomly then it effect the model
3. Reduce Information: It reduces the degree of freedom of the model

Dealing with Missing Values cont…

- The most common way to deal with the missing values is "*fillna()*" method

Dealing with other issues

- Data issues are not limited
- Take an example of a monetarily value, suppose fetch data from excel file the some number filed may contain the $, Rs. and other signs

[]: Df['RawSalary'] = df['RawSalary'].str.replace('$','')

- Now convert the data type

[]: df['RawSalary'] = df['RawSalary'].astype('float')

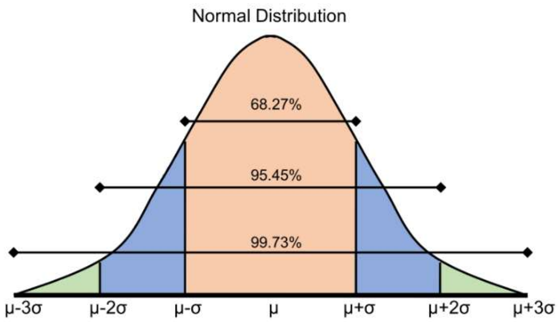
Feature Engineering
Data Distribution

Data Distribution

- Important consideration before built-in the machine learning model the distribution of the underlying data
- There are several algorithm can be used to get how the data is distributed and how different features interact each other
- Like all models beside tree-based model required the data in the same scale
- Feature engineering allows to manipulate the data

Distribution assumption

- Except tree-based model all model required the data is in normal distribution like bell shape shown in figure



Data Distribution cont…

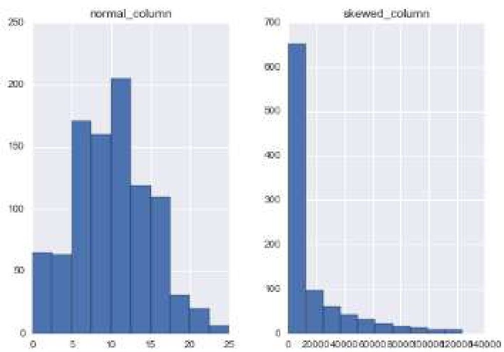
- The property of the normal distribution is that 68.27% of data lies between one standard deviation of the mean, 95.45% lies two standard deviation of the mean, and 99.73% lies three standard deviation of the mean

Observing your data by python

[]: import matplotlib as plt

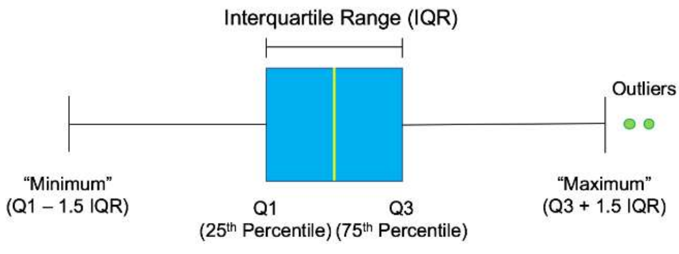
[]: df.hist()

[]: plt.show()



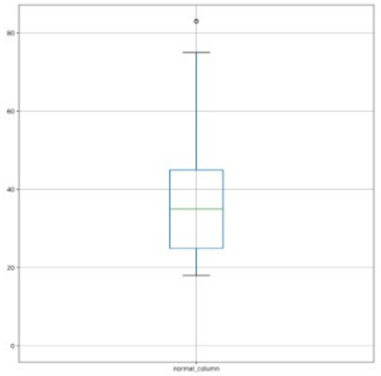
Delivering deeper with box plot

- Any point outside the box is outlier



Delivering deeper with box plot cont…

- []: df[['column_1']].boxplot()
- []: plt.show()



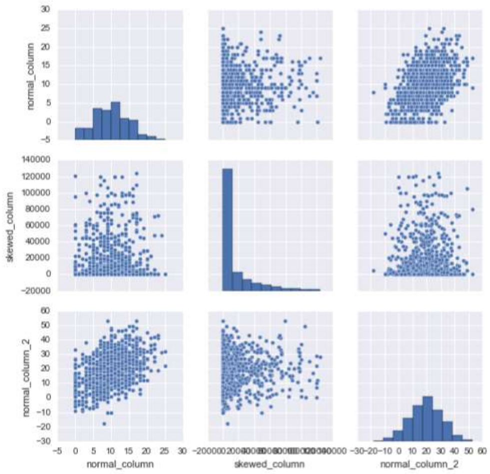
Pairing Distribution

[]: import seaborn as sns

[]: sns.pairplort(df)

The above python code shows the correlation or association to each other

[]: df.describible()



Feature Engineering
Scaling and Transformation

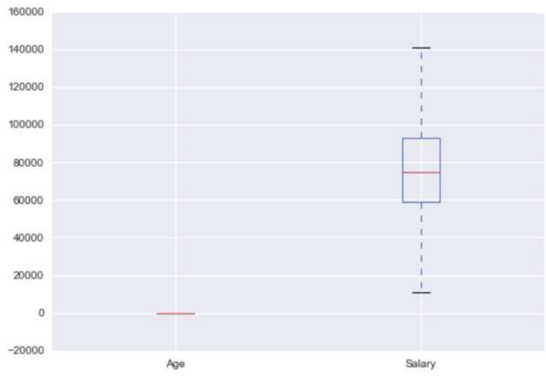
Scaling

- Most machine learning model required the data on same scale
- It is difficult to campus the salary value often in thousand with ages
- Similar scale is measuring need to rescale and bring all data on same scale

Scaling cont…

- Most machine learning model required the data on same scale
- It is difficult to campus the salary value often in thousand with ages
- Similar scale is measuring nee to rescale and bring all data o same scale

Scaling data

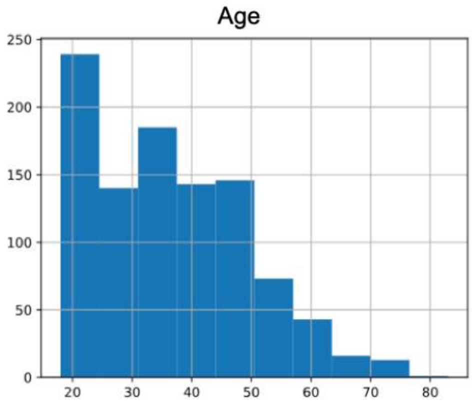


Scaling…

- There are different types of scaling technique, two of them are
  1. Min-max scaling
  2. Standardization

Min-max scaling

- It is also called linear scaling
- The python code is



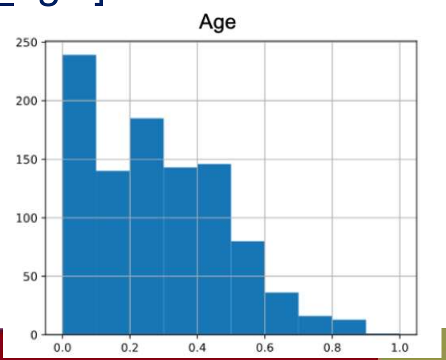
Min-max scaling cont…

[]: from sklearn.preprocessing import MinMaxScaler

[]: Scaler = MinMaxScaler()

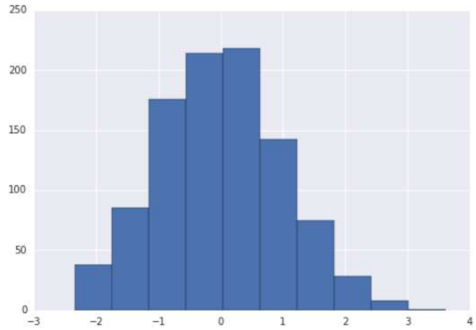
[]: Scaler.fit(df[[‘age’]])

[]: df[‘normalized_age’]= scaler.transform(df[[‘age]])



Standardization

- Standardization finds the mean of the data and center of the distribution around it
- Finding the standard deviation from mean this has no max and min values



Standardization cont…

[]: from sklearn.preprocessing import standardScaler

[]: scaler = StandardScaler()

[]: scaler.fit(Df[['age']])

[]: df['standardScaler_col'] = scaler.transform(df[['age']])

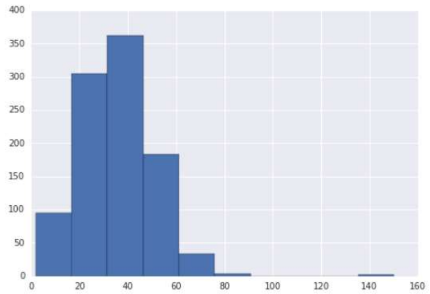
Removing Outliers

- After transformation data is very skewed because outliers which exist in data

What are the Outliers

- Outliers which is far away from majority of the data

What are outliers?



Removing Outliers

How to delete it

- There are several methods to remove the outliers some of them are

1. Quantile based approach
2. Standard deviation based deletion

Outlier- Quantile based approach for removing

- We can remove top 5% of data
- But if there is no outlier the removing the actual data may remove

[]: q_cutoff = df['col_name'].quantile(0.95)

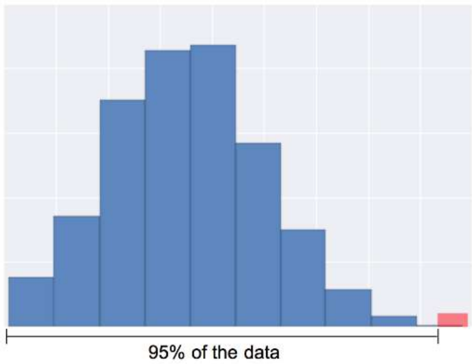
[]: mask = df['col_name'] < q_cutoff

[]: trimmed_off = df[mask]

Outlier- Quantile based approach for removing

- d

Quantile based detection



Outlier- Standard deviation based deletion

- This method support to remove only genuine outlier values

[]: mean = df['co_name'].mean()

[]: std = df['col_name'].std()

[]: cut_off = std*3

[]: lower,upper = mean – cut_off, mean + cutoff

[]: new_df = df[(df['col_name'] < upper & (df['col_name'] > lower)]

Outlier- Standard deviation based deletion

- d

Standard deviation based detection