IBM PROJECT

PHASE 3 : DEVELOPMENT (PART 1)

TOPIC : MEASURE ENERGY CONSUMPTION MODEL BY LOADING AND PRE-PROCESSING THE DATASET

MEASURE ENERGY CONSUMPTION

INTRODUCTION :

- Artificial intelligence (AI) can be used to measure energy consumption. AI can enhance data collected from smart meters, plugs, and thermostats. AI can also:
- Identify inefficiencies in production lines
- Recommend adjustments to improve energy efficiency
- Determine energy needs
- Automatically redistribute energy resources
- Determine when excess energy should be stored or sold back to the grid

GIVEN DATA SET FOR TIME SERIES FORECASTING WITH MACHINE LEARNING :

```
Score = np.sqrt(mean_squared_error(test[‘PJME_MW’], test[‘prediction’]))
print(f’RMSE Score on Test set: {score:0.2f}’)
RMSE Score on Test set: 3721.75
```

Calculate Error Look at the worst and best predicted days

```
test[‘error’] = np.abs(test[TARGET] – test[‘prediction’])
test[‘date’] = test.index.date
test.groupby([‘date’])[‘error’].mean().sort_values(ascending=False).head(10)
```

```
      date
2016-08-13    12839.597087
2016-08-14    12780.209961
2016-09-10    11356.302979
2015-02-20    10965.982259
2016-09-09    10864.954834
2018-01-06    10506.845622
2016-08-12    10124.051595
2015-02-21     9881.803711
2015-02-16     9781.552246
2018-01-07     9739.144206
```

NECESSARY STEPS TO FOLLOW:

1.IMPORT LIBRARIES

Start by uploading the necessary libraries

program :

```
# Python program using Pandas for
# arranging a given set of data
# into a  table
# importing pandas as pd
import pandas as pd
data = {“country”: [“Brazil”, “Russia”, “India”, “China”, “South Africa”],
       “capital”: [“Brasilia”, “Moscow”, “New Delhi”, “Beijing”, “Pretoria”],
       “area”: [8.516, 17.10, 3.286, 9.597, 1.221],
    “population”: [200.4, 143.5, 1252, 1357, 52.98]
data_table = pd.DataFrame(data)
print(data_table)
```

2.LOAD THE DATASET :

- To load a dataset into a Pandas Data frame, you can use the read_csv() function. For example, you can use the following code to load a world energy consumption dataset

Program:

```
From matplotlib import pyplot [1]

dataset = read_csv('household_power_consumption.csv',  header=0,  infer_datetime_format=True,
parse_dates=['datetime'],  index_col=['datetime'])  [1]

days = [x for x in range(1, 20)] [1]

for i in range(len(days)):  [1]

day = '2007-01-' + str(days[i]) [1]

result = dataset[day] [1]

pyplot [1]
```

3.EXPLORATORY DATA ANALYSIS :

- Perform EDA to understand your data better. This includes checking for missing values ,exploring the Device's statistics And visualising it to Identify Patterns.

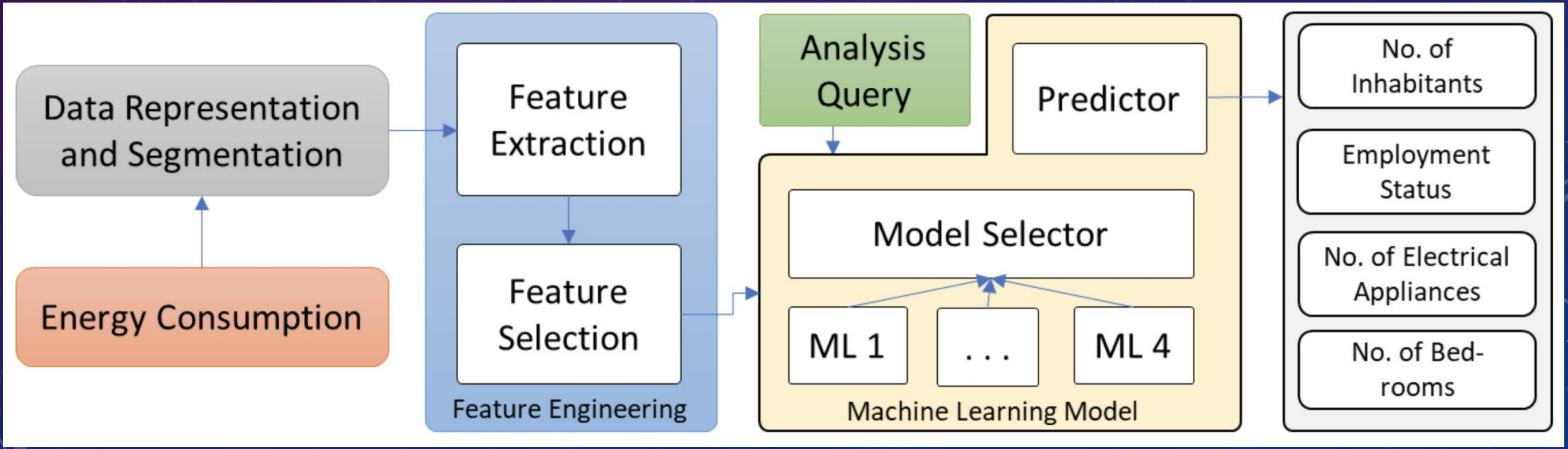
Program:

```
#checking for missing values

Print (df.isnull()sum())

#explore statistics

Print (df.described())

#visualize the data(eg.Histograms,scatter plots etc.)
```

4.FEATURE ENGINEERING:

- Depending on your dataset you may need to create new features or Transforming existing one. This can involve one-hot encoding categorical variables, handling date/time data, or scaling numerical features.

Diagram:



SPLIT THE DATA:

- Split your dataset into training and testing sets. This helps you evaluate your model's performance later.

Program::

```
# Importing the turicreate Library
import turicreate as tc
# Now Loading the data
data=tc.Sframe("data.csv")
 # Turicreate has a library named as random
# split that will the data randomly among the train,test
# Dev will be part of test set and we will split that data later.
Train_data_set,test_data=data.random_split(.8,
```

FEATURE SCALING:

- Apply feature scaling to normalise your data, ensuring that all features have similar scales, standardization (scaling to mean=0 and Std=1) is a common choice.

Program:

```
From sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(df)
scaled_df = pd.DataFrame(scaled_data,
columns=df.columns
scaled_df.head()
```

IMPORTANCE OF LOADING AND PRE-PROCESSING DATASET :

- Loading a dataset is important for fast access and analysis. It can also improve data accuracy and speed up time to insights.
- In machine learning, loading a dataset is a significant preliminary step to build any deep learning model. You have to load and pre-process the data that you want to train your model using.
- Data pre-processing is essential before its actual use. Data pre-processing is the concept of changing the raw data into a clean data set. The dataset is pre-processed in order to check missing values, noisy data, and other inconsistencies before executing it to the algorithm. Data must be in a format appropriate for ML.

CHALLENGES INVOLVING LOADING AND PRE-PROCESSING DATASET:

Missing data:

Datasets often have missing values, which can adversely affect the performance of machine learning models.

Categorical Data:

Machine learning algorithms typically work with numerical data, so categorical variables need to be transformed into a numerical format.

Scaling and Normalization:

Features in the dataset may have different scales, which can affect the performance of algorithms like k-nearest neighbours or gradient descent-based methods.

Feature Engineering:

Determining which features are relevant and how to create new meaningful features from the existing ones can be complex.

SOLUTIONS TO OVERCOME THE CHALLENGES INVOLVING LOADING AND PRE-PROCESSING DATASET:

Missing Data:

Techniques such as imputation (filling missing values using statistical methods), removing rows or columns with missing values, or using advanced imputation algorithms can address this challenge.

Categorical Data:

Techniques like one-hot encoding (for nominal data) or label encoding (for ordinal data) can be applied to convert categorical variables into a suitable numerical format.

scaling and normalisation:

Scaling methods like Min-Max scaling or standardization (z-score normalization) can be applied to bring features to a similar scale, ensuring fair comparison between t

Feature engineering:

domain knowledge and experimentation play a significant role in feature engineering. Techniques like PCA (Principal Component Analysis) or domain-specific transformations can help create relevant features.

1.LOADING THE DATASET:

Program:

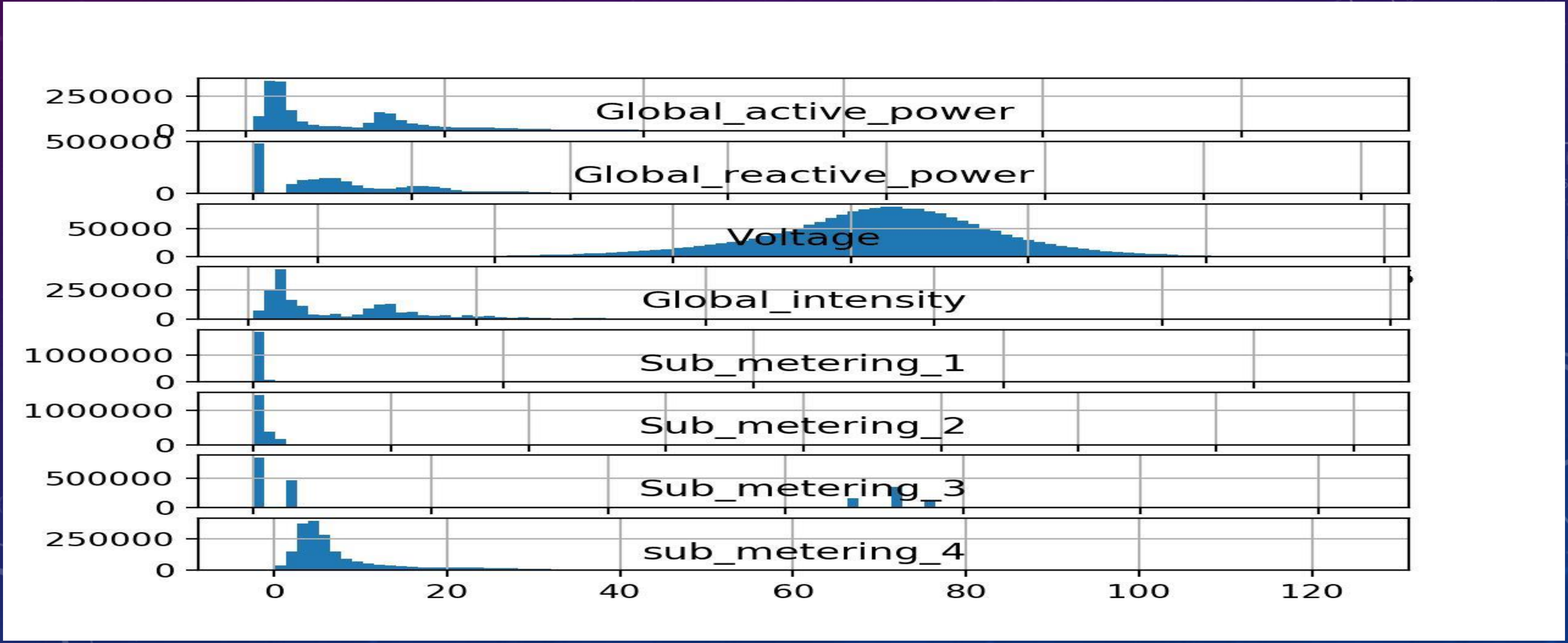
Input:

```
: monthly line plots
from pandas import read_csv
from matplotlib import pyplot
# load the new file
dataset = read_csv('household_power_consumption.csv', header=0, infer_datetime_format=True, parse_dates=['datetime'], index_col=['datetime'])
# plot active power for each year
months = [x for x in range(1, 13)]
pyplot.figure()
for i in range(len(months)):
        # prepare subplot
        ax = pyplot.subplot(len(months), 1, i+1)
        # determine the month to plot
        month = '2007-' + str(months[i])
        # get all observations for the month
        result = dataset[month]
        # plot the active power for the month
        pyplot.plot(result['Global_active_power'])
        # add a title to the subplot
        pyplot.title(month, y=0, loc='left')
pyplot.show()
```

OUTPUT:

```
2016-08-13   12839.597087
2016-08-14   12780.209961
2016-09-10   11356.302979
2015-02-20   10965.982259
2016-09-09   10864.954834
2018-01-06   10506.845622
2016-08-12   10124.051595
2015-02-21    9881.803711
2015-02-16    9781.552246
2018-01-07    9739.144206
```

EXAMPLE GRAPH:(HOUSEHOLD ELECTRICITY CONSUMPTION)



2.PREPROCESSING THE DATASET:

- Data pre-processing is used to convert raw data into a clear format. Raw data consist of missing values, noisy data, and raw data may be text, image, numeric values, etc.
- By the above definition, we understood that transforming unstructured data into a structured form is called data pre-processing. If the unstructured data is used in machine learning models to analyse or to predict, the prediction will be false because unstructured data contains missing values and unwanted data. So for good prediction, the data need to be pre-processed

VISUALIZATION AND PRE-PROCESSING THE DATASET:

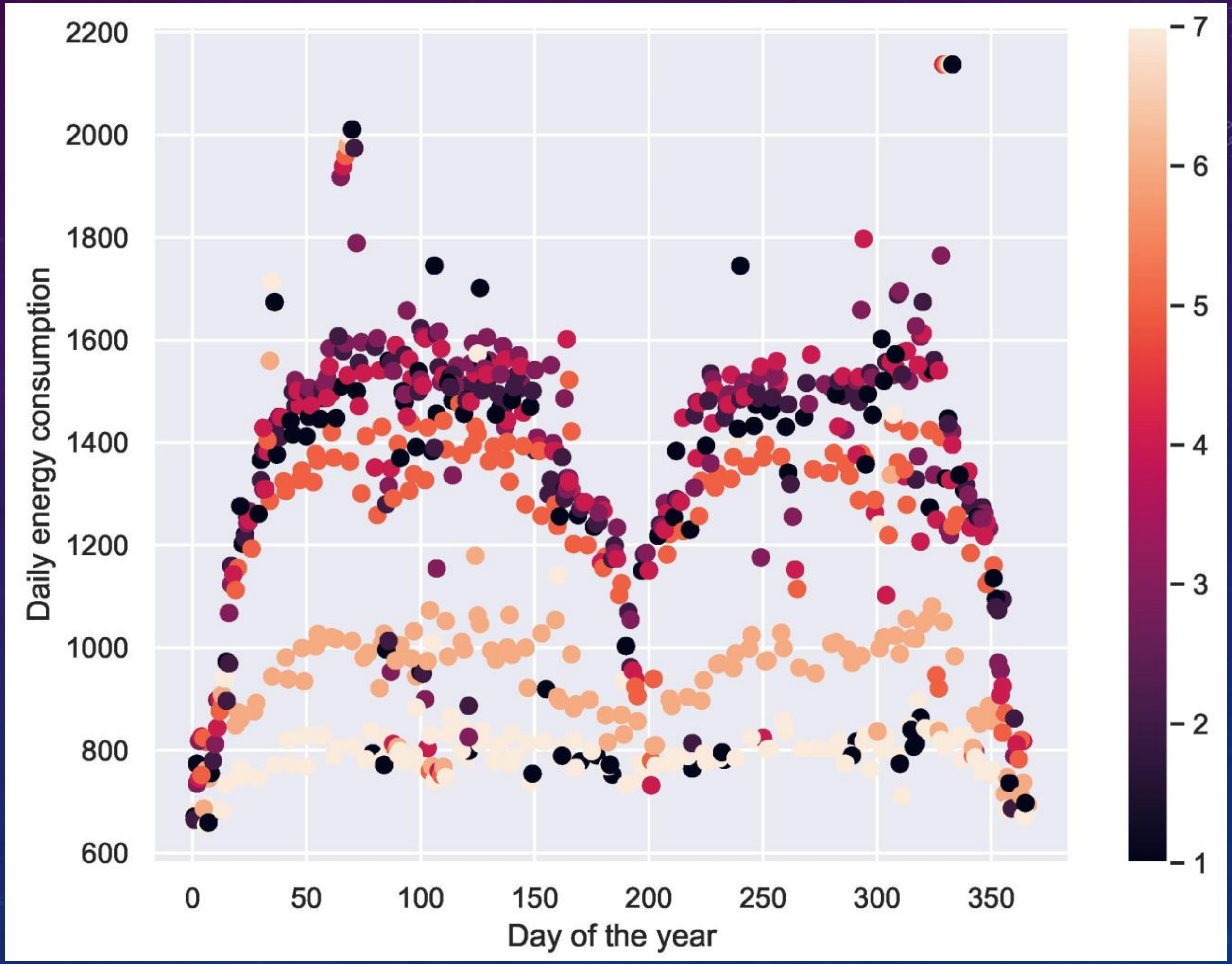
```
Import pandas
import matplotlib.pyplot as plt
data = ‘iris_df.csv’
names = [‘sepal-length’, ‘sepal-width’, ‘petal-length’, ‘petal-width’, ‘class’]
dataset = pandas.read_csv(data, names=names)
dataset.plot(kind=‘box’, subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()
```

Input:

```
Energy.count()
```

Output:

```
day           829
energy_sum    829
LCLid         829
dtype: int64
```



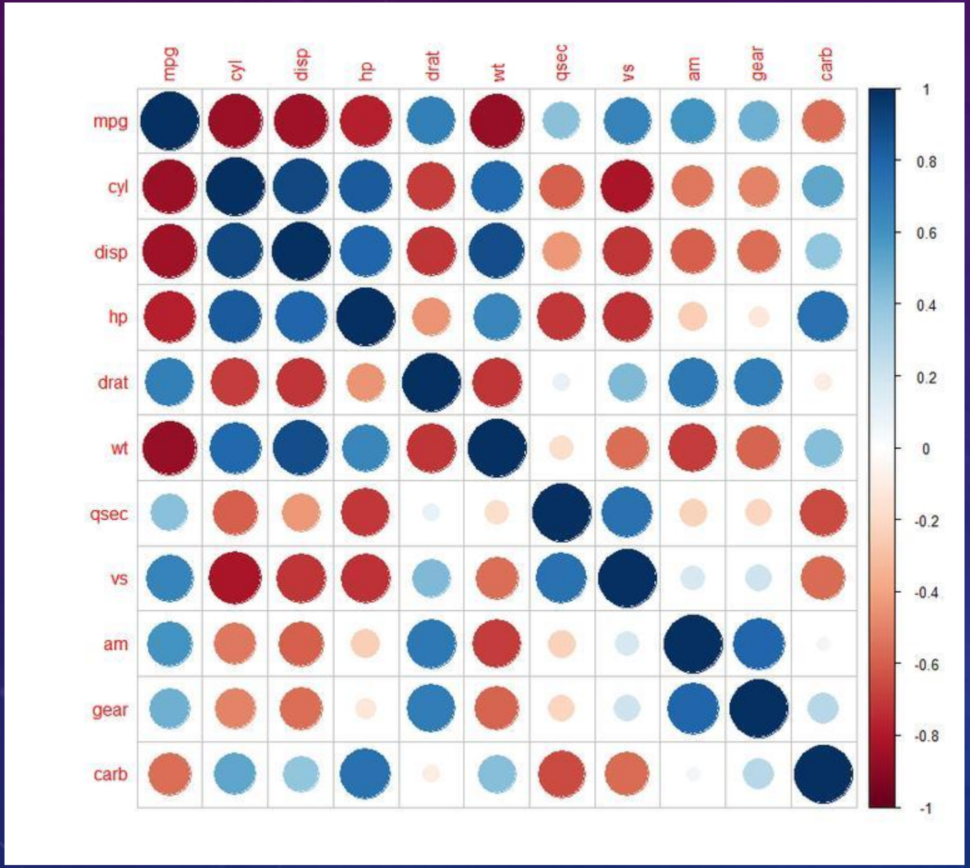
VISUALIZATION CORRELATION :

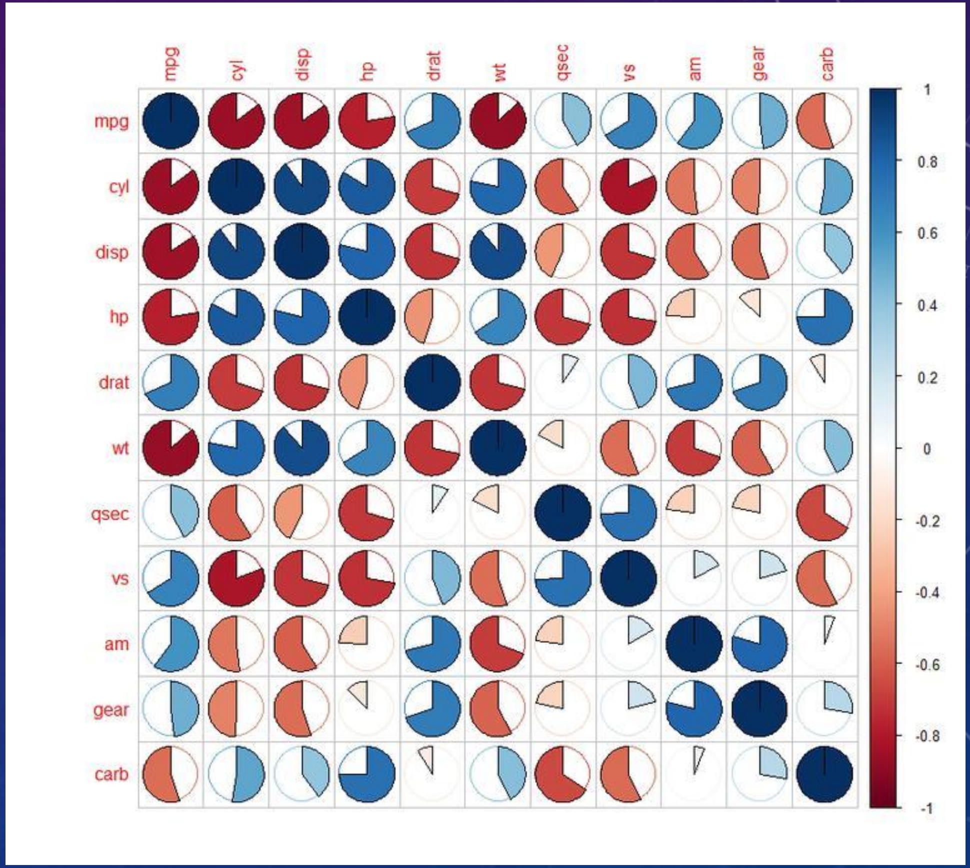
```
# Chronogram in R
# including the required packages
library(corrplot)
head(mtcars)
Head(mtcars)
                  mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

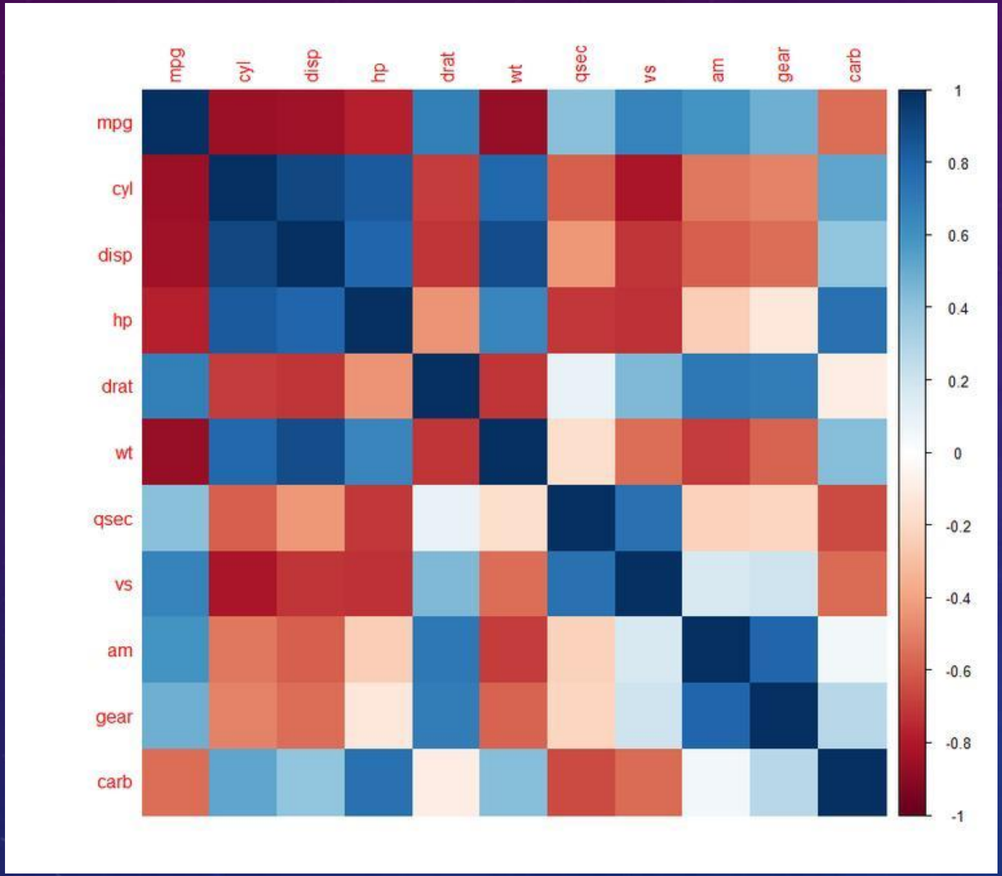
input:

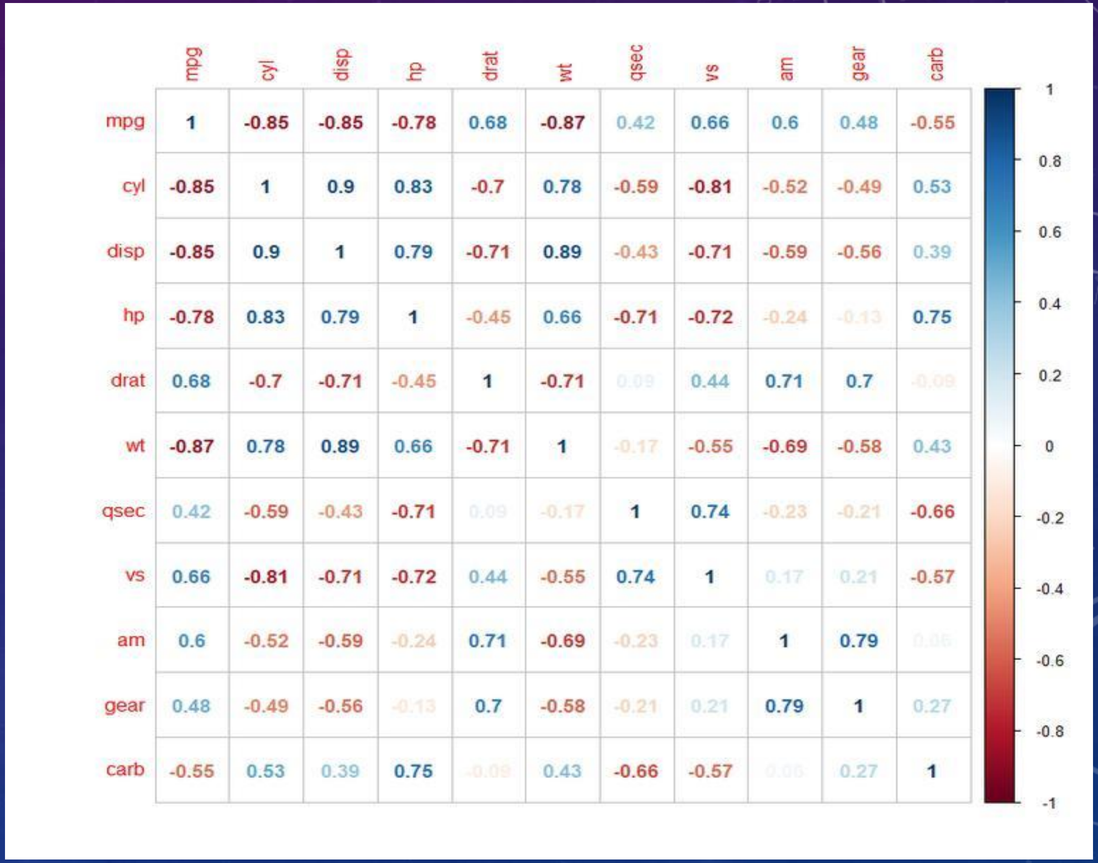
```
# Correlogram in R
# required packages
library(corrplot)
head(mtcars)
#correlation matrix
M<-cor(mtcars)
head(round(M,2))
#visualizing correlogram
#as circle
corrplot(M, method=“circle”)
#as pie
corrplot(M, method=“pie”)
# as colour
corrplot(M, method=“color”)
# as number
correlate(M, method=“number”)
```

OUTPUT:


mpg
cyl
disp
hp
drat
wt
qsec
vs
am
gear
carb
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1


mpg
cyl
disp
hp
drat
wt
qsec
vs
am
gear
carb
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1





	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
mpg	1	-0.85	-0.85	-0.78	0.68	-0.87	0.42	0.66	0.6	0.48	-0.55
cyl	-0.85	1	0.9	0.83	-0.7	0.78	-0.59	-0.81	-0.52	-0.49	0.53
disp	-0.85	0.9	1	0.79	-0.71	0.89	-0.43	-0.71	-0.59	-0.56	0.39
hp	-0.78	0.83	0.79	1	-0.45	0.66	-0.71	-0.72	-0.24	-0.13	0.75
drat	0.68	-0.7	-0.71	-0.45	1	-0.71	0.09	0.44	0.71	0.7	-0.09
wt	-0.87	0.78	0.89	0.66	-0.71	1	-0.17	-0.55	-0.69	-0.58	0.43
qsec	0.42	-0.59	-0.43	-0.71	0.09	-0.17	1	0.74	-0.23	-0.21	-0.66
vs	0.66	-0.81	-0.71	-0.72	0.44	-0.55	0.74	1	0.17	0.21	-0.57
am	0.6	-0.52	-0.59	-0.24	0.71	-0.69	-0.23	0.17	1	0.79	0.06
gear	0.48	-0.49	-0.56	-0.13	0.7	-0.58	-0.21	0.21	0.79	1	0.27
carb	-0.55	0.53	0.39	0.75	-0.09	0.43	-0.66	-0.57	0.06	0.27	1

SOME COMMON DATA PRE-PROCESSING TASKS INCLUDE:

Data profiling:

Data profiling is the process of examining, analysing and reviewing data to collect statistics about its quality. It starts with a survey of existing data and its characteristics. Data scientists identify data sets that are pertinent to the problem at hand, inventory its significant attributes, and form a hypothesis of features that might be relevant for the proposed analytics or machine learning task.

Data cleansing:

The aim here is to find the easiest way to rectify quality issues, such as eliminating bad data, filling in missing data or otherwise ensuring the raw data is suitable for feature engineering.

Data reduction:

Raw data sets often include redundant data that arise from characterizing phenomena in different ways or data that is not relevant to a particular ML, AI or analytics task. Data reduction uses techniques like principal component analysis to transform the raw data into a simpler form suitable for particular use cases.

Data transformation:

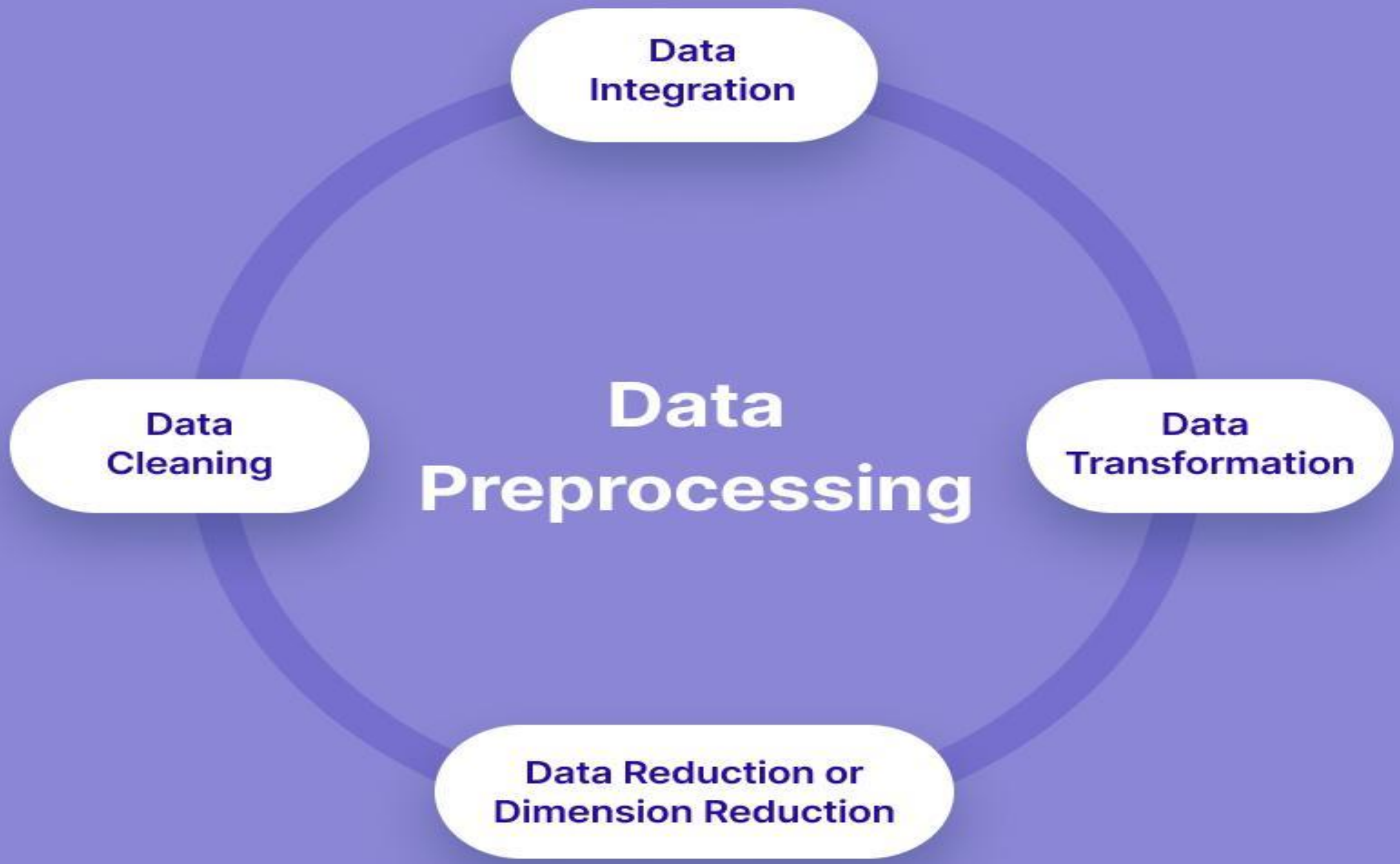
Here, data scientists think about how different aspects of the data need to be organized to make the most sense for the goal. This could include things like structuring unstructured data, combining salient variables when it makes sense or identifying important ranges to focus on.

Data enrichment:

In this step, data scientists apply the various feature engineering libraries to the data to effect the desired transformations. The result should be a data set organized to achieve the optimal balance between the training time for a new model and the required compute.

Data validation:

At this stage, the data is split into two sets. The first set is used to train a machine learning or deep learning model. The second set is the testing data that is used to gauge the accuracy and robustness of the resulting model. This second step helps identify any problems in the hypothesis used in the cleaning and feature engineering of the data. If the data scientists are satisfied with the results, they can push the pre-processing task to a data engineer who figures out how to scale it for production. If not, the data scientists can go back and make changes to the way they implemented the data cleansing and feature engineering steps.


Data Integration
Data Cleaning
Data Preprocessing
Data Transformation
Data Reduction or Dimension Reduction

CODE FOR PRE-PROCESSING A DATASET IN PYTHON:

Import libraries:

```
import pandas as pd

from sklearn.pre-processing import Imputer
```

Read the dataset:

```
dataset = pd.read_csv(‘Data.csv’)
```

Pre-process the data:

```
X = dataset.iloc[:, :-1].values

imputer = imputer.fit(X[:, 1:3])
```