**SVKM’s NMIMS**

**Mukesh Patel School of Technology Management & Engineering**

Program: B.Tech\MBA.Tech

**Course: Machine Learning**

**Experiment No.03**

PART A

(PART A : TO BE REFFERED BY STUDENTS)

**A.1 Aim:** Apply feature scaling and normalization techniques to transform raw features into representations suitable for machine learning model and implement techniques for feature selection.

**A.2 Prerequisite:**

Python Programming, Pandas library, Numpy Library, MatplotLib, Seaborn Library

**A.3 Outcome:**

**After successful completion of this experiment students will be able to:**

* 1. Understand and implement different feature scaling and normalization techniques
  2. Understand and implement different feature selection techniques

**A.4 Theory:**

**Tasks:**

**Feature Scaling:**

Scaling / Feature Scaling, is a technique in which we scale our data before employing an algorithm so that all the features contribute equally to the result.

Normalization and Standardization are two types of Feature Scaling.

One of the reasons that it's easy to get confused between scaling and normalization is because the terms are sometimes used interchangeably and, to make it even more confusing, they are very similar! In both cases, you're transforming the values of numeric variables so that the transformed data points have specific helpful properties. The difference is that:

in scaling, you're changing the range of your data, while in normalization, you're changing the shape of the distribution of your data.

By scaling your variables, you can help compare different variables on equal footing.

**Scaling:**

This means that you're transforming your data so that it fits within a specific scale, like 0-100 or 0-1. You want to scale data when you're using methods based on measures of how far apart data points are, like support vector machines (SVM) or k-nearest neighbors (KNN).

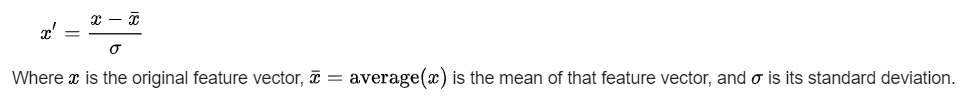
In features, we have 2 components magnitude i.e value and the other is a unit that gives information in which measure was measured. A dataset consists of many features with different magnitudes and units.

Most of the machine learning algorithms use a distance between 2 data points for computation. If left alone, these algorithms only take in the magnitude of features neglecting the units. The results would vary greatly between different units, 5kg and 5000gms. Technically 5Kg and 5000Gms are the same. The features with high magnitudes will weigh in a lot more in the distance calculations than features with low magnitudes. This is hiding the original pattern of data. To suppress this effect, we need to bring all features to the same level of magnitudes. This can be achieved by scaling.

**Scaling Techniques:**

1. **Z-score Normalization (Standardization):**

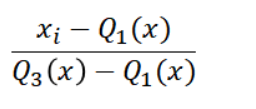
This method is widely used for normalization in many machine learning algorithms (e.g., support vector machines, logistic regression, and artificial neural networks). The general method of calculation is to determine the distribution [mean](https://en.wikipedia.org/wiki/Mean) and [standard deviation](https://en.wikipedia.org/wiki/Standard_deviation) for each feature. Next we subtract the mean from each feature. Then we divide the values (mean is already subtracted) of each feature by its standard deviation.



1. �′=�−�¯�**Robust Scaler:**

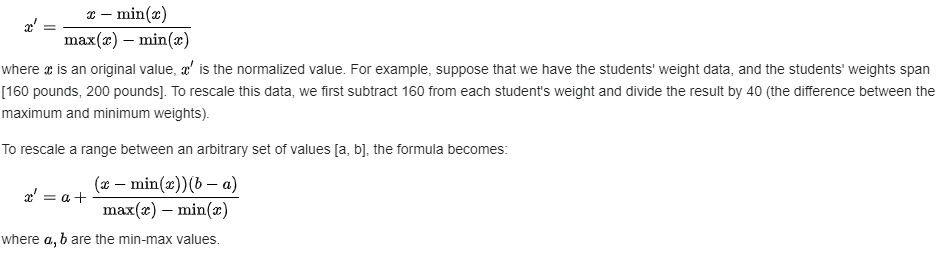
Robust Scaler algorithms scale features that are robust to outliers. The method it follows is almost similar to the MinMax Scaler but it uses the interquartile range (rather than the min-max used in MinMax Scaler). The median and scales of the data are removed by this scaling algorithm according to the quantile range.

It, thus, follows the following formula:



1. **Min-Max Scaling (Min-Max Normalization)**

Also known as min-max scaling or min-max normalization, rescaling is the simplest method and consists in rescaling the range of features to scale the range in [0, 1] or [−1, 1]. Selecting the target range depends on the nature of the data. The general formula for a min-max of [0, 1] is given as



**Feature Normalization:**

Scaling just changes the range of your data. Normalization is a more radical transformation. The point of normalization is to change your observations so that they can be described as a normal distribution.

Normal distribution: Also known as the "bell curve", this is a specific statistical distribution where a roughly equal observations fall above and below the mean, the mean and the median are the same, and there are more observations closer to the mean. The normal distribution is also known as the Gaussian distribution.

In general, you'll only want to normalize your data if you're going to be using a machine learning or statistics technique that assumes your data is normally distributed. Some examples of these include t-tests, ANOVAs, linear regression, linear discriminant analysis (LDA) and Gaussian naive Bayes. The methods that can be used to normalize are called the Box-Cox Transformation, Log Transformation etc..

**Feature Selection:**

We all may have faced the problem of identifying the important features from a set of given data and removing the irrelevant or less important features which do not contribute much to our decision making in order to achieve better accuracy for our model.

In machine learning and statistics, feature selection, also known as variable selection, attribute selection or variable subset selection, is the process of reducing the number of input variables when developing a predictive model. Feature selection techniques are used for several reasons:

It reduces model complexity by dropping some irrelevant features. Helps ML algorithm to train a model faster. Redcution of dimensionality helps in avoid overfitting. In this notebook i will be discussing 3 common techniques used for feature selection which are easy to implement and will give you a good results based on problem. Following are the feature selection techniques:

* + - 1. Univariate Selection
      2. Feature Importance
      3. Correlation Matrix with Heatmap

**Task 1:**

Generate 1000 data points randomly drawn from an exponential distribution.

Apply min-max scaling on the data set using inbuilt library and plot histogram for original data and scaled data.

Write your own function to define min-max scalar and apply it on data generated. Plot histogram for original data and scaled data.

Apply z-score normalization technique on the data generated. Plot histogram for original data and scaled data.

Apply robust scalar on the data generated. Plot histogram for original data and scaled data.

**Task 2:**

1. Upload mobile dataset and convert train data set into dataframe.
2. Apply chi2 feature selection technique of sklearn library to identify most important features of the mobile data set.
3. Apply XG Boost classifier to find the feature importance score of the mobile data set and identify most important features.
4. Apply correlation matrix on mobile dataset and plot heat map. State your inference.

PART B

(PART B : TO BE COMPLETED BY STUDENTS)

***(Students must submit the soft copy as per following segments within two hours of the practical.)***

|  |  |
| --- | --- |
| Roll No. N052 | Name: Pratyush Kumar |
| Class : MBA Tech CE (div. D) | Batch : B2 |
| Date of Experiment: 13-01-2024 | Date of Submission: 13-01-2024 |
| Grade : |  |

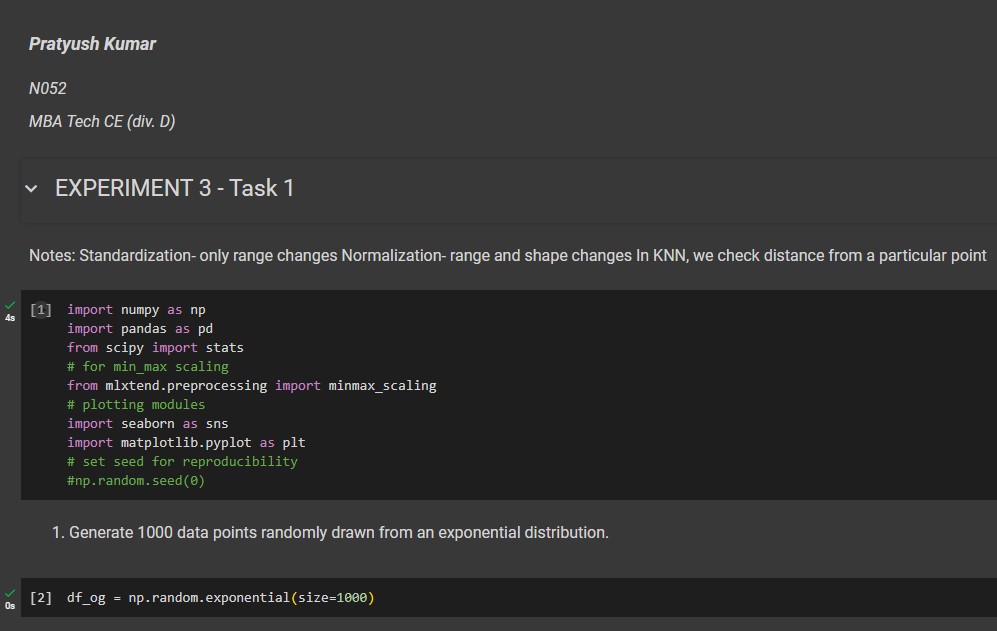
**B.1 Task 1**

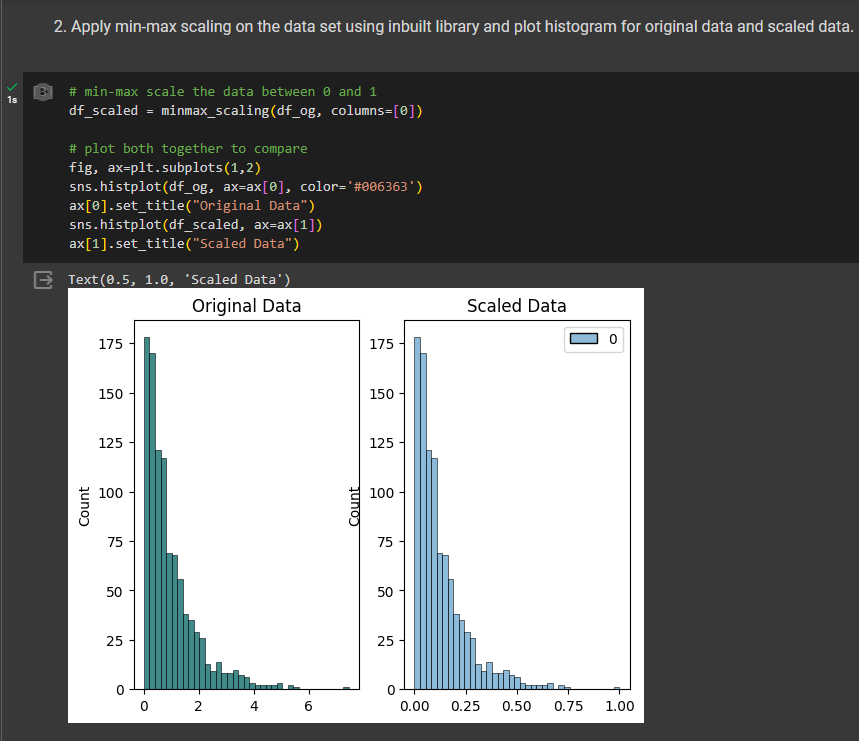
**Colab link:** <https://colab.research.google.com/drive/1gfPjTXA5_f_iFMKCXV_C08tY_MYU0o8a?usp=sharing>

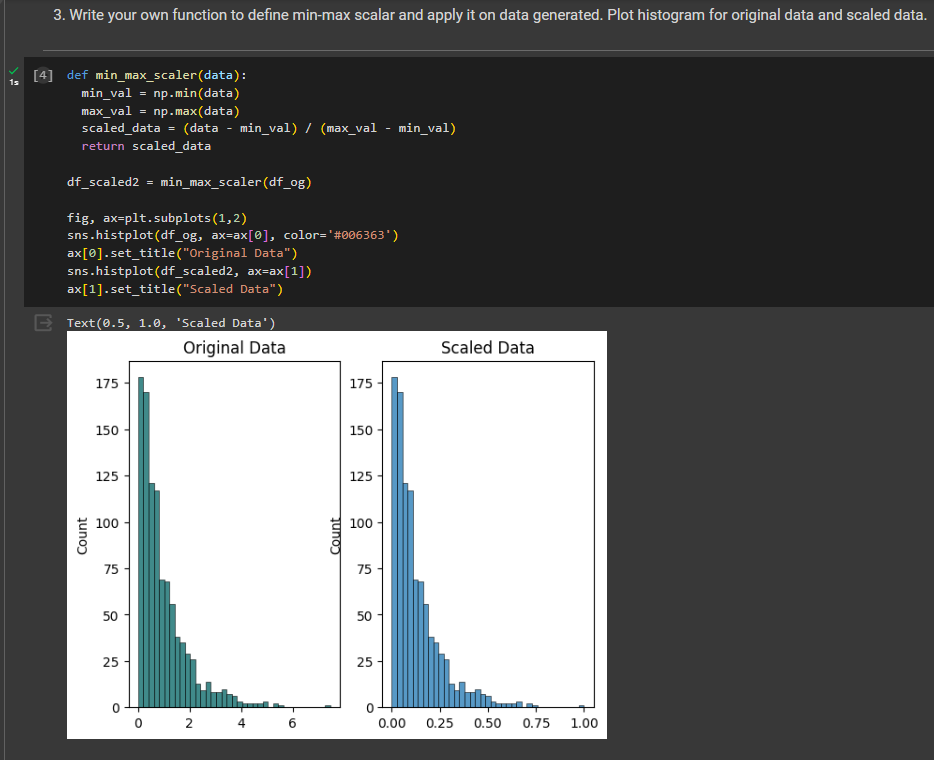
* **Source Code**

*"""  
 \* This file contains code snippets for implementing feature scaling and normalization techniques   
 \* ML-E3-Task1  
 \*  
 \* Original file is located at: https://colab.research.google.com/drive/1gfPjTXA5\_f\_iFMKCXV\_C08tY\_MYU0o8a  
 \* @author Pratyush Kumar (github.com/pratyushgta)  
"""*"""  
## EXPERIMENT 3 - Task 1  
  
Notes:  
Standardization- only range changes  
Normalization- range and shape changes  
In KNN, we check distance from a particular point  
"""  
  
import numpy as np  
import pandas as pd  
from scipy import stats  
# for min\_max scaling  
from mlxtend.preprocessing import minmax\_scaling  
# plotting modules  
import seaborn as sns  
import matplotlib.pyplot as plt  
# set seed for reproducibility  
#np.random.seed(0)  
  
"""1. Generate 1000 data points randomly drawn from an exponential distribution."""  
  
df\_og = np.random.exponential(size=1000)  
  
"""2. Apply min-max scaling on the data set using inbuilt library and plot histogram for original data and scaled data."""  
  
# min-max scale the data between 0 and 1  
df\_scaled = minmax\_scaling(df\_og, columns=[0])  
  
# plot both together to compare  
fig, ax=plt.subplots(1,2)  
sns.histplot(df\_og, ax=ax[0], color='#006363')  
ax[0].set\_title("Original Data")  
sns.histplot(df\_scaled, ax=ax[1])  
ax[1].set\_title("Scaled Data")  
  
"""3. Write your own function to define min-max scalar and apply it on data generated. Plot histogram for original data and scaled data."""  
  
def min\_max\_scaler(data):  
 min\_val = np.min(data)  
 max\_val = np.max(data)  
 scaled\_data = (data - min\_val) / (max\_val - min\_val)  
 return scaled\_data  
  
df\_scaled2 = min\_max\_scaler(df\_og)  
  
fig, ax=plt.subplots(1,2)  
sns.histplot(df\_og, ax=ax[0], color='#006363')  
ax[0].set\_title("Original Data")  
sns.histplot(df\_scaled2, ax=ax[1])  
ax[1].set\_title("Scaled Data")  
  
"""4. Apply z-score normalization technique on the data generated. Plot histogram for original data and scaled data."""  
  
from scipy.stats import zscore  
  
# Applying Z-score normalization using scipy.stats.zscore  
df\_standard = zscore(df\_og)  
  
plt.figure(figsize=(10, 5))  
  
plt.subplot(1, 2, 1)  
plt.hist(df\_og, bins=20, color='#006363', edgecolor = 'black')  
plt.title('Original Data')  
plt.xlabel('Values')  
plt.ylabel('Frequency')  
  
plt.subplot(1, 2, 2)  
plt.hist(df\_standard, bins=20, color='#630063', edgecolor = 'black')  
plt.title('Z-score Normalized Data')  
plt.xlabel('Standardized Values')  
plt.ylabel('Frequency')  
  
"""5. Apply robust scalar on the data generated. Plot histogram for original data and scaled data."""  
  
from sklearn.preprocessing import RobustScaler  
  
df\_og\_copy = df\_og.copy()  
  
# OG data is in 1D. For robust scaler, we need 2D data  
df\_og\_copy = df\_og\_copy.reshape(-1, 1)  
  
# Applying RobustScaler for robust normalization  
scaler = RobustScaler()  
df\_scaler = scaler.fit\_transform(df\_og\_copy)  
  
# Plotting og and scaled data  
plt.figure(figsize=(10, 5))  
  
plt.subplot(1, 2, 1)  
plt.hist(df\_og\_copy, bins=20, color='#006363', edgecolor='black')  
plt.title('Original Data')  
plt.xlabel('Values')  
plt.ylabel('Frequency')  
  
plt.subplot(1, 2, 2)  
plt.hist(df\_scaler, bins=20, color='#006363', edgecolor='black')  
plt.title('Scaler Data')  
plt.xlabel('Values')  
plt.ylabel('Frequency')

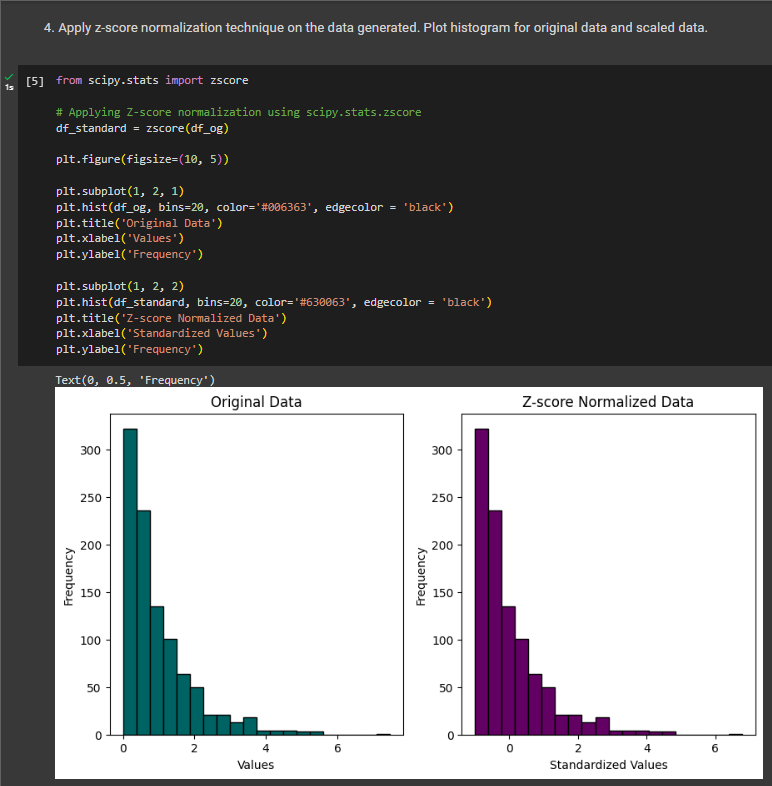
* **Input/ Output**

****

****

****

****

****

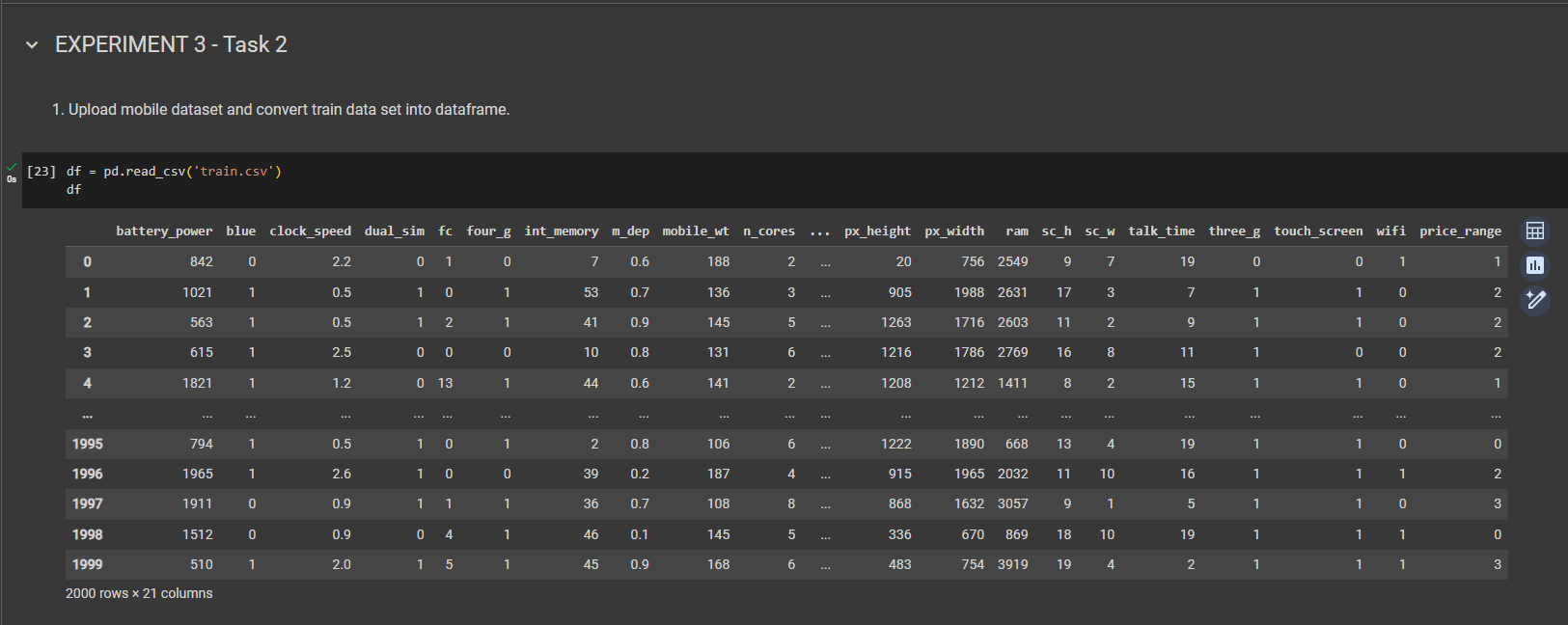
**B.2 Task 2**

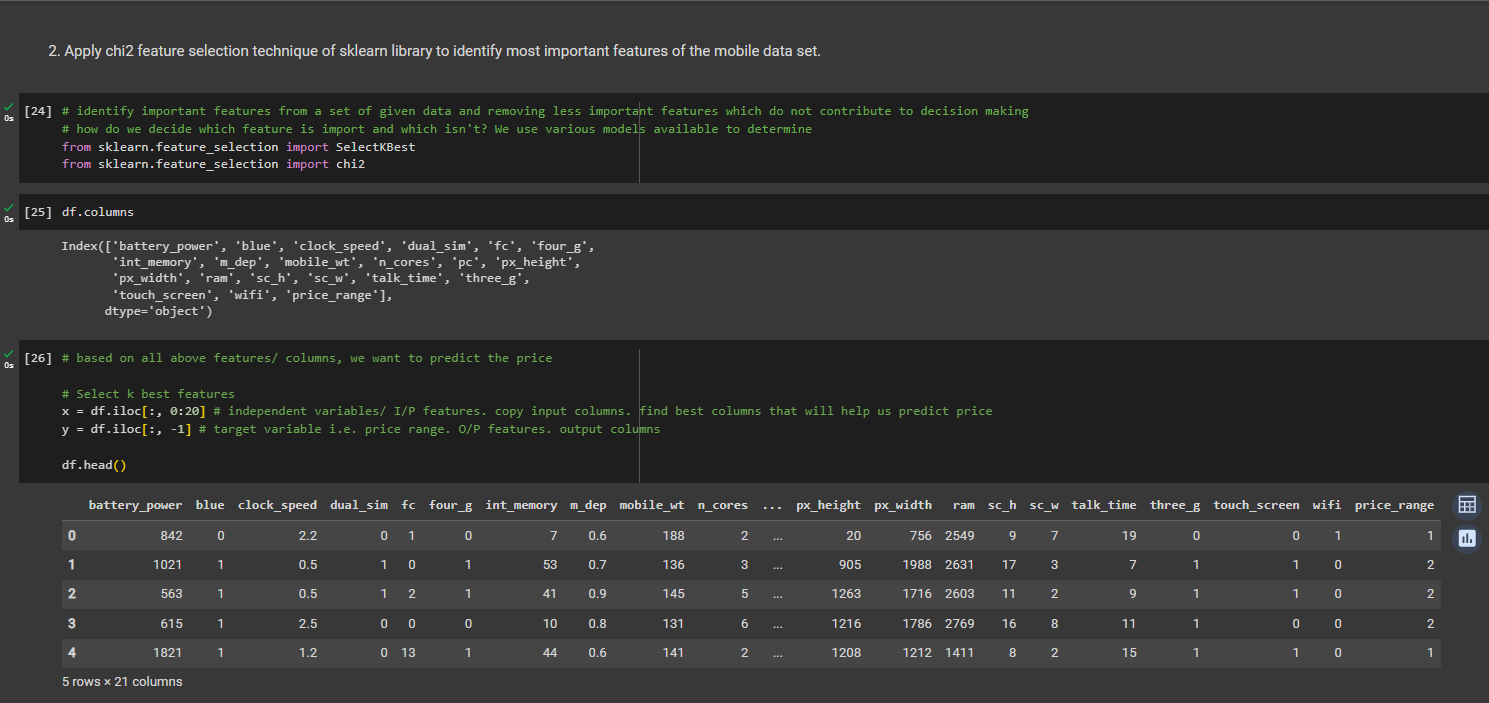
**Colab link:** <https://colab.research.google.com/drive/1gfPjTXA5_f_iFMKCXV_C08tY_MYU0o8a?usp=sharing>

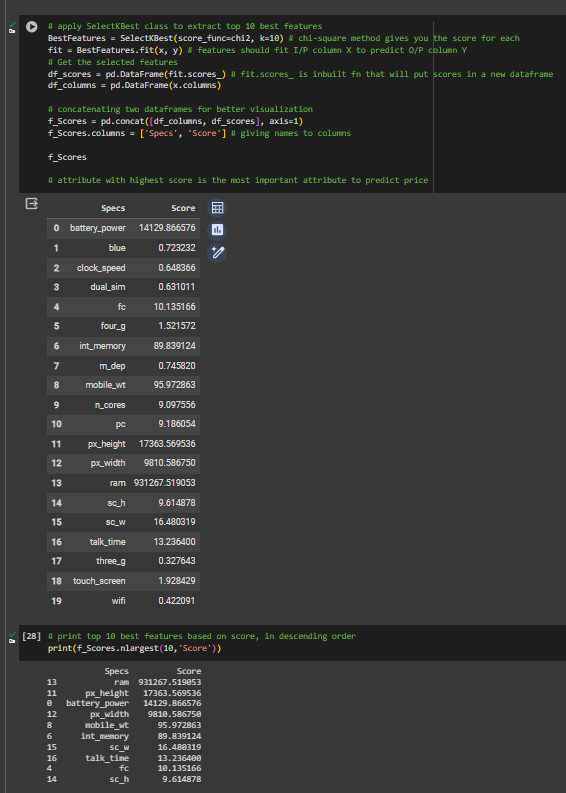
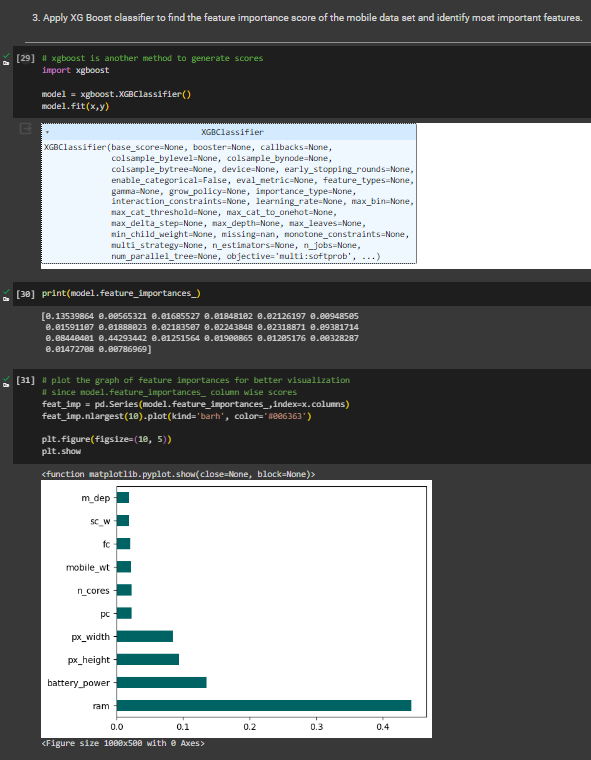
* **Source Code**

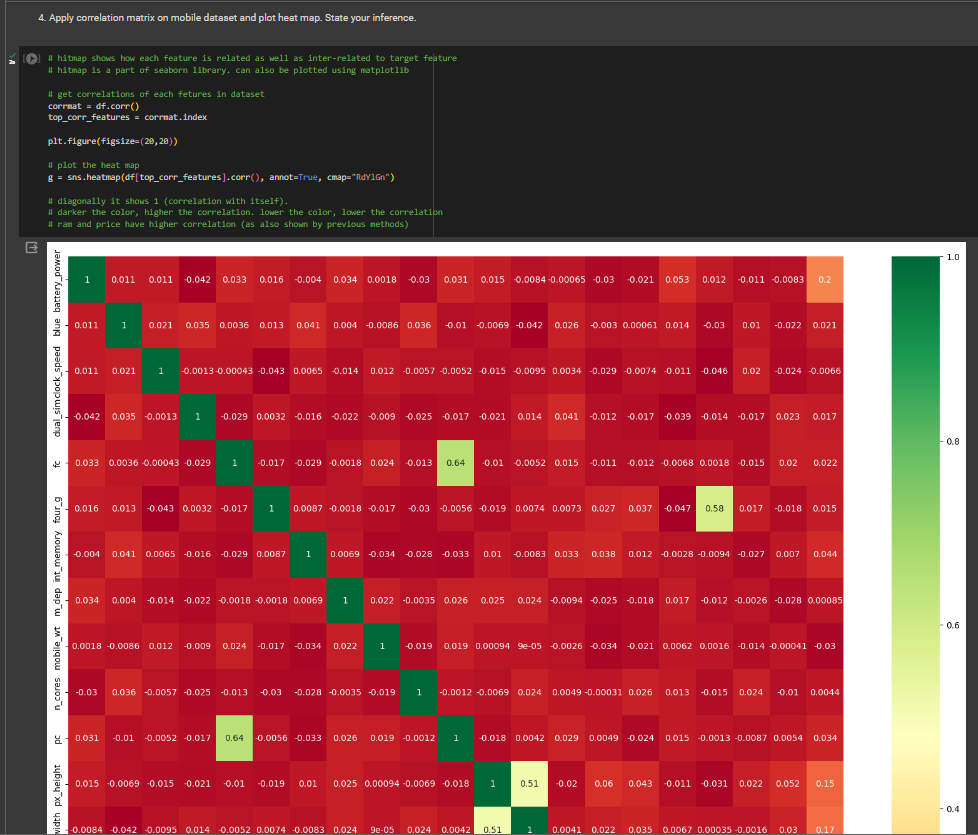
*"""  
 \* This file contains code snippets to implement different feature selection techniques in mobile/ train dataset  
 \* ML-E3-Task2  
 \*  
 \* Original file is located at: https://colab.research.google.com/drive/1gfPjTXA5\_f\_iFMKCXV\_C08tY\_MYU0o8a  
 \* @author Pratyush Kumar (github.com/pratyushgta)  
"""*"""## EXPERIMENT 3 - Task 2  
  
1. Upload mobile dataset and convert train data set into dataframe.  
"""  
  
import numpy as np  
import pandas as pd  
from scipy import stats  
# for min\_max scaling  
from mlxtend.preprocessing import minmax\_scaling  
# plotting modules  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
df = pd.read\_csv('train.csv')  
df  
  
"""2. Apply chi2 feature selection technique of sklearn library to identify most important features of the mobile data set."""  
  
# identify important features from a set of given data and removing less important features which do not contribute to decision making  
# how do we decide which feature is import and which isn't? We use various models available to determine  
from sklearn.feature\_selection import SelectKBest  
from sklearn.feature\_selection import chi2  
  
df.columns  
  
# based on all above features/ columns, we want to predict the price  
  
# Select k best features  
x = df.iloc[:, 0:20] # independent variables/ I/P features. copy input columns. find best columns that will help us predict price  
y = df.iloc[:, -1] # target variable i.e. price range. O/P features. output columns  
  
df.head()  
  
# apply SelectKBest class to extract top 10 best features  
BestFeatures = SelectKBest(score\_func=chi2, k=10) # chi-square method gives you the score for each  
fit = BestFeatures.fit(x, y) # features should fit I/P column X to predict O/P column Y  
# Get the selected features  
df\_scores = pd.DataFrame(fit.scores\_) # fit.scores\_ is inbuilt fn that will put scores in a new dataframe  
df\_columns = pd.DataFrame(x.columns)  
  
# concatenating two dataframes for better visualization  
f\_Scores = pd.concat([df\_columns, df\_scores], axis=1)  
f\_Scores.columns = ['Specs', 'Score'] # giving names to columns  
  
f\_Scores  
  
# attribute with highest score is the most important attribute to predict price  
  
# print top 10 best features based on score, in descending order  
print(f\_Scores.nlargest(10,'Score'))  
  
"""3. Apply XG Boost classifier to find the feature importance score of the mobile data set and identify most important features."""  
  
# xgboost is another method to generate scores  
import xgboost  
  
model = xgboost.XGBClassifier()  
model.fit(x,y)  
  
print(model.feature\_importances\_)  
  
# plot the graph of feature importances for better visualization  
# since model.feature\_importances\_ column wise scores  
feat\_imp = pd.Series(model.feature\_importances\_,index=x.columns)  
feat\_imp.nlargest(10).plot(kind='barh', color='#006363')  
  
plt.figure(figsize=(10, 5))  
plt.show  
  
"""4. Apply correlation matrix on mobile dataset and plot heat map. State your inference."""  
  
# hitmap shows how each feature is related as well as inter-related to target feature  
# hitmap is a part of seaborn library. can also be plotted using matplotlib  
  
# get correlations of each fetures in dataset  
corrmat = df.corr()  
top\_corr\_features = corrmat.index  
  
plt.figure(figsize=(20,20))  
  
# plot the heat map  
g = sns.heatmap(df[top\_corr\_features].corr(), annot=True, cmap="RdYlGn")  
  
# diagonally it shows 1 (correlation with itself).  
# darker the color, higher the correlation. lower the color, lower the correlation  
# ram and price have higher correlation (as also shown by previous methods)

* **Input/ Output**

****

****

****

****

**B.3 Conclusion:**

*(Students must write the conclusion in their own words.)*

Implemented various feature scaling and normalization techniques such as z-score, min-max scaling, and decimal scaling to transform raw features into representations suitable for machine learning model. Also implemented techniques for feature selection, including filter methods (variance analysis, correlation) and wrapper methods (recursive feature elimination).