1

Devanshu Surana
PC-23, 1032210755
Panel C, Batch C1

DEC Lab Assignment 3

Aim: Preprocess data using python

Problem Statement: Data Pre-processing using Python (Part II)

Objectives:
Data Integration
Data Redundancy and correlation Analysis
Tuple Duplication
Data Transformation
Normalization Min-max, z-score
Data smoothening - Binning Methods (on dataset such as csv/xls file)
Data Reduction
PCA method.

Theory:
1. Pearson Correlation : NumPy and SciPy implementation
Pearson correlation is a statistical method used to measure the linear relationship betn two continous variables.
NumPy and SciPy, python libraries provide function like 'numpy' 'corrcoef' and 'scipystats' pearson' to calculate pearson correlation coefficients betn data arrays.

These functions helps access know strongly and in what direction two variables are correlated with values ranging from -1 to 1.

2. Pearson Correlation: Pandas Implementation
Pandas is a popular data manipulation library, offers the pandas. ~~Dataforce~~ Dataframe .corr method to compute pearson correlation coefficients within a Dataframe.
We can use this method to quickly analyze and understand the relationships betn multiple variable in a structured data set.

3. Visualization of Correlation:
To visualize correlation between variables, you can create correlation matrices and use various visualization techniques.

4. Heatmaps of correlation Matrices
Heatmaps are the common way to visualize correlation matrices where colors represent the strength and direction of correlation.

5. Feature Normalization and their techniques.
It is the process of scaling data to ensure all features have a similar influence on machine learning algo. Common techniques includes:
Minimax Scaling
Standardization
Log Transformation

Conclusion:

Hence we have ~~Submitted~~ studied Data preprocessing i.e Data integration, Transformation and Reduction.

FAQ's

1. Why do we need scaling?

→ Scaling is needed to ensure that all features in a dataset have a similar impact on machine learning models. it helps prevent features with larger from dominating these with smaller scales

2. Benefits and Techniques of Binning in Python.

→ Binning discretizes continous data into intervals or bins, simplifying complex data reducing noise and aiding analysis. Techniques like equal-width and equal-frequency binning are used.

3. What is Data leakage. How to avoid any data leakage during the model testing process?

→ Data leakage happens when information from the test set unintentionally leaks into the training process, causing overly optimistic model performance. To prevent this, apply all data preprocessing and feature engineering steps consistently.

4. Which technique we should use Normalization or standardization?

→ It depends on the algorithm and the data distribution. Use standardization (z-score) for algorithms sensitive to varying scales or normalization (Min-max Scaling) for models relying on values within specific ranges

5. What are the benefits of correlation Analysis?

→ It helps identify relationships between variables allowing prediction, understanding dependencies, selecting relevant features and detecting multicollinearity in datasets.

6. What is the significance of correlation analysis?

→ Correlation analysis is significant as it helps in understanding the strength and direction of relationships betn var, aiding in decision-making, feature selection, and predicting the behaviour of related attributes in datasets.

7. What are the different kinds of correlation analysis. Discuss their strength and weakness.

→ a) Pearson Correlation Coefficient

Strengths:

1. Measures linear relationships betn continous variables.
2. Easy to interpret, with values ranging from its

weakness:

1. Assumes that the variables are normally distributed and have a linear relationship.

b) Spearman Rank Correlation Coefficient ~~st~~.

Strengths:

1. Non-parametric, meaning it doesn't rely on specific data distribution assumption.

Measures monotonic (non-linear) relationship bet^n variable.

Weakness:

1. Less sensitive to subtile linear relationship
2. Ignores specific data values, only focusing on their ranks.

c) Kendall's Tau (τ)

Strengths:

1. Also non-parametric and suitable for non-linear relationship.
2. Measures association bet^n variables ordinal data.

Weakness:

1. Less commonly used in applications compared to pearsons and spearman correlation.
2. Computationally more intensive for large dataset.

8. What are the factors that affect a correlation Analysis?

→ Sample size
- Data Distribution

and the choice of correlation coefficient can affect the results of a correlation analysis.

9. Write a short note on:

a. The correlation coefficient:

→ It quantifies the strength and direction of the linear relationship bet^n two continous variables. It ranges from -1 to 1 where -1 is negative linear relati-onship +1 is positive linear relationship and 0 is no relationship.

b. The p-value:

→ The p-value in correlation analysis measures the significance of the calculated correlation coefficient. A low p-value indicates a statically significant correlation meaning that the observed relationship is unlikely to be due to change. Conversely a high p-value suggest a weaker or non-significant correlation.

```
from sklearn import preprocessing
import numpy as np
x_array=np.array([2,3,5,6,7,4,8,7,6])
normalized_arr=preprocessing.normalize([x_array])
print(normalized_arr)
```

```
[[0.11785113 0.1767767  0.29462783 0.35355339 0.41247896 0.23570226
  0.47140452 0.41247896 0.35355339]]
```

```
from google.colab import files
uploaded=files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving housing.csv to housing.csv
```

```
from sklearn import preprocessing
import pandas as pd
housing=pd.read_csv("/content/sample_data/california_housing_train.csv")
scaler =preprocessing.MinMaxScaler()
names=housing.columns
d=scaler.fit_transform(housing)
scaler_df=pd.DataFrame(d,columns=names)
scaler_df.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_val
0	1.000000	0.175345	0.274510	0.147885	0.198945	0.028364	0.077454	0.068530	0.1070
1	0.984064	0.197662	0.352941	0.201608	0.294848	0.031559	0.075974	0.091040	0.1342
2	0.975100	0.122210	0.313725	0.018927	0.026847	0.009249	0.019076	0.079378	0.1457
3	0.974104	0.116897	0.254902	0.039515	0.052142	0.014350	0.037000	0.185639	0.1204
4	0.974104	0.109458	0.372549	0.038276	0.050435	0.017405	0.042921	0.098281	0.1041

```
scaler_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17000 entries, 0 to 16999
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           17000 non-null  float64
 1   latitude            17000 non-null  float64
 2   housing_median_age  17000 non-null  float64
 3   total_rooms         17000 non-null  float64
 4   total_bedrooms      17000 non-null  float64
 5   population          17000 non-null  float64
 6   households          17000 non-null  float64
 7   median_income       17000 non-null  float64
 8   median_house_value  17000 non-null  float64
dtypes: float64(9)
memory usage: 1.2 MB
```

```
scaler_df.isnull().sum()
```

```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
median_income         0
median_house_value    0
dtype: int64
```

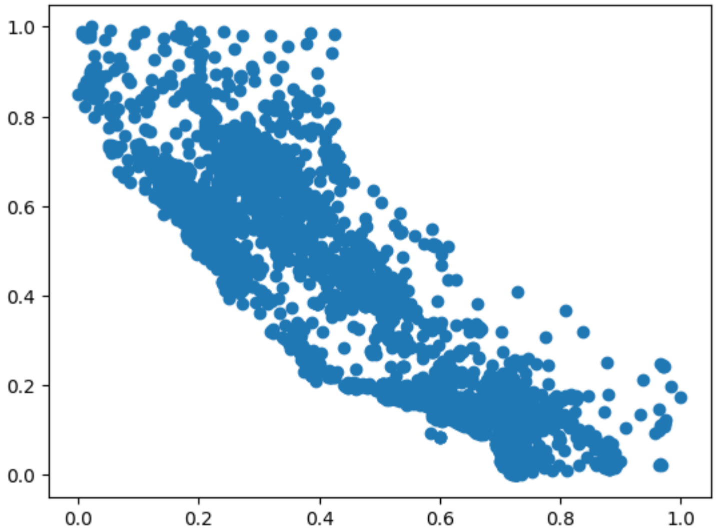
```
scaler_df.describe()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	me
count	17000.000000	17000.000000	17000.000000	17000.000000	17000.000000	17000.000000	17000.000000	17000.000000	
mean	0.476882	0.327867	0.540968	0.069637	0.083552	0.039984	0.082260	0.233354	
std	0.199718	0.227135	0.246803	0.057465	0.065410	0.032172	0.063233	0.131595	

```
import matplotlib.pyplot as plt
x=([scaler_df.longitude])
y=([scaler_df.latitude])

plt.scatter(x,y)
plt.plot()
```

```
[]
```



```
from google.colab import files
uploaded=files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving mark (1).csv to mark (1).csv

```
from google.colab import files
uploaded=files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving student (1).csv to student (1).csv

```
student=pd.read_csv("/content/student (1).csv")
mark=pd.read_csv("/content/mark (1).csv")
```

```
merged=pd.merge(mark,student,on="Student_id")
merged.head()
```

	Student_id	Mark	City	Age	Gender	Grade	Employed
0	1	95	Chennai	19	Male	1st Class	yes
1	2	70	Delhi	20	Female	2nd Class	no
2	3	98	Mumbai	18	Male	1st Class	no
3	4	75	Pune	21	Female	2nd Class	no
4	5	89	Kochi	19	Male	1st Class	no

```
merged.isnull().sum()
```

```
Student_id    0
Mark          0
City          0
Age           0
Gender        0
Grade         0
Employed      0
dtype: int64
```

```
merged.describe()
```

	Student_id	Mark	Age
count	232.000000	232.000000	232.000000
mean	116.500000	71.400862	19.896552
std	67.116814	17.116069	1.030944
min	1.000000	40.000000	18.000000
25%	58.750000	55.000000	19.000000
50%	116.500000	75.000000	20.000000
75%	174.250000	85.250000	21.000000
max	232.000000	100.000000	22.000000