3.facebook.ipynb

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv("dataset_Facebook (1).csv" , delimiter=';')

df

df.isnull().sum()

(df==0).sum()

df.describe()

# Creating Subset of a Dataframe
subset1 = df[['like', 'share']]
subset1

# Merge Datasets : The merge() operation is a method used to combine two dataframes based on one or more common columns.
# The resulting data frame contains only the rows from both dataframes with matching column.
# ex : DataFrame_name.merge(right, how='inner', on=None, left_on=None, right_on=None, left_index=False, right_index=False, sort=False, suffixes=('_x', '_y'), copy=True, indicator=False, validate=None)

subset2 = df[['Post Month', 'Post Weekday']]
subset2

merged_sub = pd.merge(subset1, subset2, how = "inner",left_on = "like", right_on = "Post Month")
merged_sub  #if no col is common in both dataframes we have to use left_on & right_on and how is optional
```

```
subset3 = df[['like', 'comment', 'Post Hour']]

subset3
```

```
merged_subset = pd.merge(merged_sub, subset3, how = "left", on = "like") # on is used when there is any col is common in both datasets

merged_subset
```

```
merged_subset.sort_values(by=['like'],ascending=False)
```

```
merged_subset.T    # merged_subset.transpose()
```

HEART

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv("heart (1).csv")
df

df.describe()

df.isnull().sum()   # df.isna().sum()

df.isnull().sum()   # df.isna().sum()

(df == 0).sum()

df.duplicated()

df = df.drop_duplicates(subset=None, keep='first', inplace=False, ignore_index=False)
# df.drop_duplicates()
# When working with multiple data sources, there are many chances for data to be incorrect,
# duplicated, or mislabeled. If data is wrong, outcomes and algorithms are unreliable, even though
# they may look correct. Data cleaning is the process of changing or eliminating garbage, incorrect,
# duplicate, corrupted, or incomplete data in a dataset.
# Handling missing values, removing duplicate entries, dropping un-necessary columns in a dataframe, changing index of a dataframe

df.duplicated().any()

df.duplicated()
```

```
# Data Integration

df1 = df[['age', 'cp']]
df2 = df[['age','chol', 'fbs']]

integrated_data = pd.merge(df1, df2, how = 'inner', on = "age")
integrated_data

# data correcting
import matplotlib.pyplot as plt
import seaborn as sns

sns.boxplot(x = df['chol'])
plt.show

Q1 = df['chol'].quantile(0.25)
Q3 = df['chol'].quantile(0.75)
IQR = Q3 - Q1
IQR

low_limit = Q1 - 1.5 * IQR
upp_limit = Q3 + 1.5 * IQR
low_limit, upp_limit

outliers_low = (df['chol'] < low_limit)
outliers_upp = (df['chol'] > upp_limit)

df = df[~(outliers_low | outliers_upp)]

sns.boxplot(x = df['chol'])
plt.show
```

```
# Data transformation

X = df.iloc[:, 1:6]
Y = df.iloc[:, 1:6]

from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, random_state = 42)

X_train.shape, X_test.shape, Y_train.shape, Y_test.shape

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

x_train_scaled = scaler.fit_transform(X_train)
x_test_scaled = scaler.transform(X_test)

# Data model building

x = df[['age','cp','chol','fbs']]
y = df[['target']]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
y_train= np.array(y_train).reshape(-1, 1)
y_test= np.array(y_test).reshape(-1, 1)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,confusion_matrix
# import seaborn as sns
# import matplotlib.pyplot as plt
model = LogisticRegression()
model.fit(x_train_scaled, y_train)

# Make predictions on the test set
y_pred = model.predict(x_test_scaled)

# Evaluate the model's accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Air.ipynb

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv('air_quality.csv', encoding='cp1252')
df

df.describe()

df.isnull().sum()

(df == 0).sum()

missing_per = (df.isnull().sum() / df.shape[0]) * 100

np.round(missing_per, decimals=2)

# we can drop columns having greater no of null values.
# we can replace null values with specific value like for numerical column we can replace null values by mean value or a value which occur frequently
# for categorical column we can replace null value by a value which occur frequently

df = df.drop(['stn_code', 'agency', 'spm', 'pm2_5'], axis = 1)

df.columns

df.isnull().sum()

df.mode()
```

```
df['sampling_date'].value_counts().idxmax()

df.sampling_date[(df.sampling_date.isna())] = '19-03-15'

df.location[(df.location.isna())] = 'Guwahati'

df.type[(df.type.isna())] = "Residential"

df.so2[(df.so2.isna())] = 2.0

df.no2[(df.no2.isna())] = 13.0

df.rspm[(df.rspm.isna())] = 55.0

df.location_monitoring_station[(df.location_monitoring_station.isna())] = 'Regional Office'

df.date[(df.date.isna())] = '2015-03-19'

df.isnull().sum()

df.duplicated().any()

df = df.drop_duplicates(subset=None, keep='first', inplace=False, ignore_index=False)

df.duplicated().any()

# Data Integration
Df

df.columns

df1 = df[['state', 'location', 'type']]
```

```
df2 = df[['so2', 'no2']]

integrated_data = pd.concat([df1, df2], axis = 1)
integrated_data

# Data correcting

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

df.dtypes

import matplotlib.pyplot as plt
import seaborn as sns

sns.boxplot(x = df['rspm'])
plt.show

Q1 = df['rspm'].quantile(0.25)
Q3 = df['rspm'].quantile(0.75)
IQR = Q3 - Q1
IQR

low_limit = Q1 - 1.5 * IQR
upp_limit = Q3 + 1.5 * IQR
low_limit, upp_limit


outliers_low = (df['rspm'] < low_limit)
outliers_upp = (df['rspm'] > upp_limit)

df = df[~(outliers_low | outliers_upp)]
```

```
sns.boxplot(x = df['rspm'])
plt.show

sns.boxplot(x = df['no2'])
plt.show

Q1 = df['no2'].quantile(0.25)
Q3 = df['no2'].quantile(0.75)
IQR = Q3 - Q1
IQR

low_limit = Q1 - 1.5 * IQR
upp_limit = Q3 + 1.5 * IQR
low_limit, upp_limit

outliers_low = (df['no2'] < low_limit)
outliers_upp = (df['no2'] > upp_limit)
df = df[~(outliers_low | outliers_upp)]

sns.boxplot(x = df['no2'])
plt.show

df.dtypes

# Data Transform

from sklearn.preprocessing import LabelEncoder
col_label= ['sampling_date','location','type','location_monitoring_station','date','state']
encoder = LabelEncoder()

for col in df.columns:
    df[col] = encoder.fit_transform(df[col])
df

# label = le.fit_transform(df['state'])
# df.drop("state", axis=1, inplace=True)
# df["state"] = label
# label

# plt.figure(figsize=(16,5))
# plt.subplot(1,2,1)
# sns.distplot(df['state'])
# plt.subplot(1,2,2)
# sns.boxplot(df['state'])

df.dtypes
```

Heat1.ipynb

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
df = pd.read_csv("heart (1).csv")
df


df.isnull().sum()


df.duplicated().any()


df = df.drop_duplicates(subset=None, keep='first', inplace=False, ignore_index=False)


df.duplicated().any()


(df == 0).sum()


sns.lineplot(df)


plt.figure(figsize=(14,8))
plt.title("presence of heart disease in the patient")
sns.lineplot(df)


sns.barplot(y='age',x='cp',hue='target',data=df);


plt.figure(figsize=(10,6))
sns.heatmap(df.corr(),cmap = 'YlGnBu', annot = True)


df1 = df.iloc[:10]
fig, ax = plt.subplots(1, 1, figsize=(12, 7))
bf = np.zeros(len(df1), dtype=int)
for col in df1.columns:
    ax.bar(df1.index, df1[col], width=0.6, bottom=bf, label=col)
    bf += df1[col]
```

```
ax.set_title('Stacked Barplot', loc='left', fontsize=12, fontweight='bold')
ax.legend()
plt.show()

# temp_df = df[['age','cp','chol','fbs','oldpeak']]?
# plt.scatter(df['age'],df['trestbps'])
# plt.title('age vs trestbps')
# plt.xlabel('age')
# plt.ylabel('trestbps')

sns.scatterplot(df,x='chol',y='trestbps',hue='age')

plt.figure(figsize=(15,10))
for i,col in enumerate(temp_df.columns,1):
   plt.subplot(4,3,i)
   plt.title(f"Distribution of {col} Data")
   sns.histplot(df[col],kde=True)
   plt.tight_layout()
   plt.plot()

sns.boxplot(x = df['chol'])
plt.show



fig, ax = plt.subplots()
ax.violinplot(data, showmeans=False, showmedians=True)
ax.set_title('violin plot')
xticklabels = ['age', 'chol', 'cp']
ax.set_xticks([1,2,3])
ax.set_xticklabels(xticklabels)
ax.yaxis.grid(True)
plt.show()
```

```
sns.boxplot(df,x='sex',y='age',hue='fbs')
```

```
sns.violinplot(df,x ='sex', y ='age',hue='fbs')
```

Tip.ipynb

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv("tip.csv")
df
```

```
df.isnull().sum()

df.duplicated().any()

df = df.drop_duplicates(subset=None, keep='first', inplace=False, ignore_index=False)

df.duplicated().any()

plt.figure(figsize=(12,8))
# plt.title("presence of heart disease in the patient")
sns.lineplot(df)

from sklearn.preprocessing import LabelEncoder
col_label= ['male','smoker','day','time']
encoder = LabelEncoder()

for col in df.columns:
        df[col] = encoder.fit_transform(df[col])
df



df1 = df.iloc[:10]
fig, ax = plt.subplots(1, 1, figsize=(12, 7))
bf = np.zeros(len(df1), dtype=int)
for col in df1.columns:
    ax.bar(df1.index, df1[col], width=0.6, bottom=bf, label=col)
    bf += df1[col]

ax.set_title('Stacked Barplot', loc='left', fontsize=12, fontweight='bold')
ax.legend()
plt.show()
```

```
sns.barplot(y='day',x='time',hue='size',data=df);
```

```
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
```

```
sns.scatterplot(df,x='total_bill',y='tip',hue='time')
```

```
temp_df = df[['total_bill', 'tip', 'smoker', 'day', 'time', 'size']]
plt.figure(figsize=(15,10))
for i,col in enumerate(temp_df.columns,1):
    plt.subplot(4,3,i)
    plt.title(f"Distribution of {col} Data")
    sns.histplot(df[col],kde=True)
    plt.tight_layout()
    plt.plot()
```

```
sns.boxplot(df,x='day',y='total_bill',hue='time')
```

```
sns.violinplot(df,x ='time', y ='tip',hue='day')
```

air1.ipynb

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('air_quality.csv', encoding='cp1252')
df
```

```
df.isnull().sum()
```

```
df = df.drop(['stn_code', 'agency', 'spm', 'pm2_5'], axis = 1)
```

```
df.mode()
```

Review Scrapper

```
import requests

import bs4


request1 =requests.get('https://www.amazon.com/Fitness-Pressure-Pedometer-Waterproof-Smartwatches/dp/B0CQ56R4DS/ref=sr_1_1?_encoding=UTF8&content-id=amzn1.sym.33f8f65b-b95c-44af-8b89-e59e69e79828&dib=eyJ2IjoiMSJ9.5yqF3zrGtNJfKinuFxLEOhORze-mWDa-WCJVSyC_Si6RA7im0mC3CRrfd4GSM76LzkawLEsx3GCJRvoS3ct345zN_Y6_bgwmqMsFraM25aEhPKYhlRDoVWJEl6PUoyFWW-IpnGK5fn7vJXui8Zl0sQGMBSV6NhH1GdEVEccb5f5mieQNcq7IKPOjDiT7UvDCSqwB2cEaCQJmTnZRI9sPr9LZGOtGU0pOOvA7H9g8XnQ.KYO_lhWCvD1D5k1UJGznj7_aJmEqqhNClqkMAvl6ONU&dib_tag=se&keywords=activity+trackers+and+smartwatches&pd_rd_r=ccb755fa-3d69-461c-9282-87df4f3d4356&pd_rd_w=xTGlD&pd_rd_wg=TlB0L&pf_rd_p=33f8f65b-b95c-44af-8b89-e59e69e79828&pf_rd_r=RVKJMFVXJ4B0C1FZ6SBB&qid=1714504831&sr=8-1&th=1')


request1


request1.content


soup =bs4.BeautifulSoup(request1.text)

soup


reviews=soup.findAll('div',{'class':'a-expander-content reviewText review-text-content a-expander-partial-collapse-content'})

for review in reviews:

   print(review.get_text()+"\n\n")




rating=soup.find('span',{'class':'a-icon-alt'}).get_text()

print(rating)




individual_ratings=soup.findAll('span',{'class':'a-icon-alt'})

for indiv_rating in individual_ratings:

   print(indiv_rating.get_text()+'\n')
```

```
tags =soup.find('span',{'class':'a-size-large product-title-word-break'}).get_text()
tags
```

```
customer_name=soup.findAll('span',{'class':'a-profile-name'})
for cust in customer_name:
    print(cust.get_text()+'\n\n')
```