

Python Projects Archive:Dr Ratika Datta

1.SalesPrediction Analysis

Salesprediction - Jupyter Notebook

localhost:8888/notebooks/Salesprediction.ipynb

Jupyter Salesprediction Last Checkpoint: 04/12/2023 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Kernel starting, please wait... Trusted Python 3

In [3]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [4]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [4]:

```
data = pd.read_csv('Salesprediction.csv')
print(data.head())
```

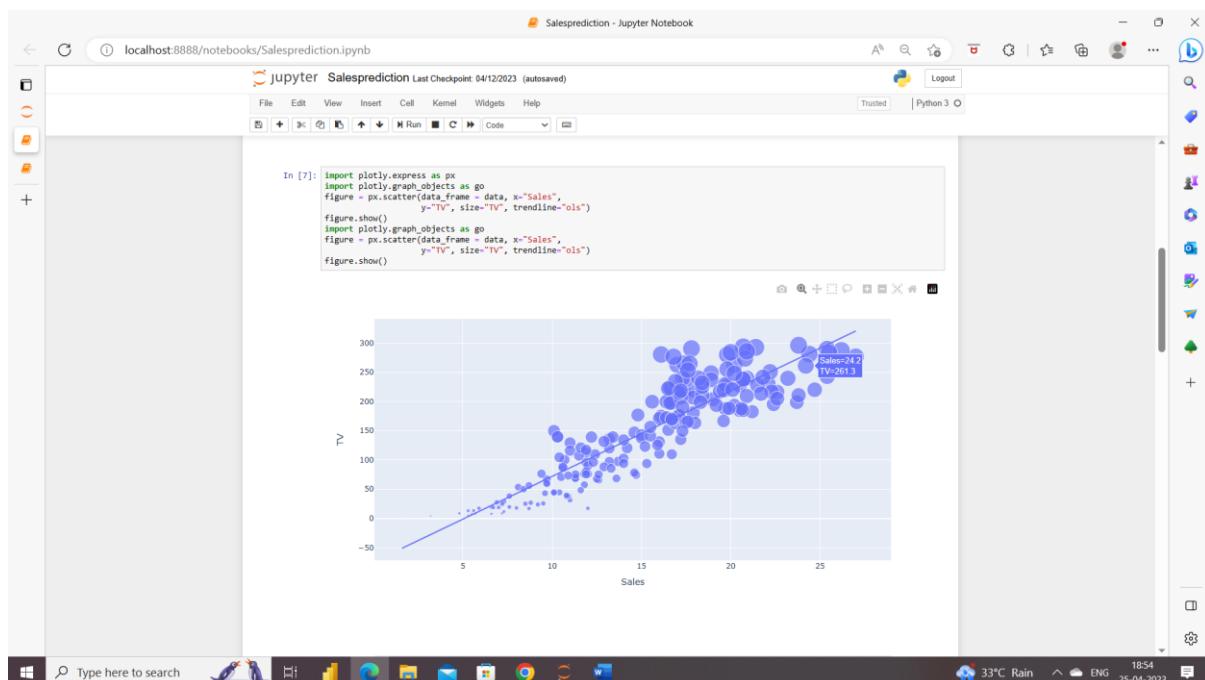
	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9

In [11]:

```
print(data.isnull().sum())
```

	TV	Radio	Newspaper
TV	0		
Radio	0		
Newspaper	0		

Type here to search 17:14 35°C Mostly cloudy 25-04-2023



Salesprediction - Jupyter Notebook

jupyter Salesprediction Last Checkpoint 04/12/2023 (autosaved)

```
In [9]: import matplotlib.pyplot as plt
import seaborn as sns

In [10]: plt.style.use('seaborn-whitegrid')
plt.figure(figsize=(12, 10))
sns.heatmap(data.corr())
plt.show()
```

```
In [11]: x = np.array(data.drop(['Sales'], 1))
y = np.array(data['Sales'])
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2, random_state=42)
model = LinearRegression()
```

35°C Mostly cloudy 17:17 25-04-2023

2. Fake News Predictor

Fakenewsdetector - Jupyter Notebook

jupyter Fakenewsdetector Last Checkpoint 3 hours ago (autosaved)

```
In [1]: import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB

data = pd.read_csv("news.csv")
print(data.head())
   Unnamed: 0                                     title \
0      8476    You Can Smell Hillary's Fear
1      3608  Watch The Exact Moment Paul Ryan Committed Pol...
2      1042      Kerry to go to Paris in gesture of sympathy
3      875  Bernie supporters on Twitter erupt in anger ag...
4      1042  The Battle of New York: Why This Primary Matters

   label
0      FAKE
1      FAKE
2      REAL
3      FAKE
4      REAL
```

```
In [2]: x = np.array(data["title"])
y = np.array(data["label"])

cv = CountVectorizer()
x = cv.fit_transform(x)

In [3]: xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2, random_state=42)
model = MultinomialNB()
model.fit(xtrain, ytrain)
print(model.score(xtest, ytest))

0.8074131002857786
```

```
In [4]: news_headline = "CA Exams 2021: Supreme Court asks ICAI to extend opt-out option for July exams, final order tomorrow"
data = cv.transform([news_headline]).toarray()
print(model.predict(data))
['REAL']

In [5]: news_headline = "You doing can cure Corona Virus"
data = cv.transform([news_headline]).toarray()
print(model.predict(data))
['FAKE']
```

35°C Mostly cloudy 17:21 25-04-2023

3. Travel Insurance Prediction:

Jupyter Notebook - TravelInsuranceprediction

```
In [5]: import pandas as pd  
data = pd.read_csv("travelinsurance.csv")  
data.head()  
  
Out[5]:
```

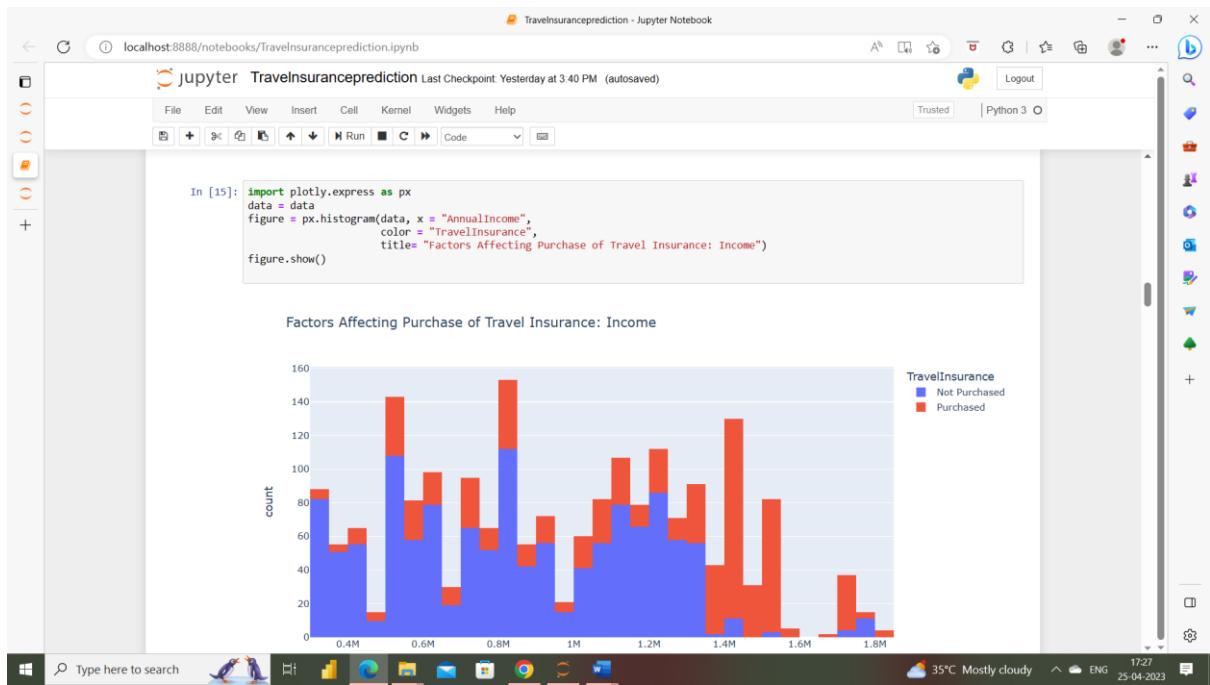
	Age	Employment Type	GraduateOrNot	AnnualIncome	FamilyMembers	ChronicDiseases	FrequentFlyer	EverTravelledAbroad	TravelInsurance
0	31	Government Sector	Yes	400000	6	1	No	No	0
1	31	Private Sector/Self Employed	Yes	1250000	7	0	No	No	0
2	34	Private Sector/Self Employed	Yes	500000	4	1	No	No	1
3	28	Private Sector/Self Employed	Yes	700000	3	1	No	No	0
4	28	Private Sector/Self Employed	Yes	700000	8	1	Yes	No	0

```
In [37]: from sklearn.metrics import classification_report, accuracy_score, f1_score  
  
In [13]: data1 = data.copy()  
data1
```

```
Out[13]:
```

	Age	Employment Type	GraduateOrNot	AnnualIncome	FamilyMembers	ChronicDiseases	FrequentFlyer	EverTravelledAbroad	TravelInsurance
0	31	Government Sector	Yes	400000	6	1	No	No	0
1	31	Private Sector/Self Employed	Yes	1250000	7	0	No	No	0
2	34	Private Sector/Self Employed	Yes	500000	4	1	No	No	1
3	28	Private Sector/Self Employed	Yes	700000	3	1	No	No	0
4	28	Private Sector/Self Employed	Yes	700000	8	1	Yes	No	0
...
1982	33	Private Sector/Self Employed	Yes	1500000	4	0	Yes	Yes	1
1983	28	Private Sector/Self Employed	Yes	1750000	5	1	No	Yes	0
1984	28	Private Sector/Self Employed	Yes	1150000	6	1	No	No	0

Windows Taskbar: Type here to search, 35°C Mostly cloudy, 17:25, 25-04-2023



TravelInsuranceprediction - Jupyter Notebook

localhost:8888/notebooks/TravelInsuranceprediction.ipynb

jupyter TravelInsuranceprediction Last Checkpoint: a few seconds ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
In [46]: data.head(10)
import numpy as np
data["GraduateOrNot"] = data["GraduateOrNot"].map({"No": 0, "Yes": 1})
data["FrequentFlyer"] = data["FrequentFlyer"].map({"No": 0, "Yes": 1})
data["EverTravelledAbroad"] = data["EverTraveledAbroad"].map({"No": 0, "Yes": 1})
data["TravelInsurance"] = data["TravelInsurance"].map({"Not Purchased": 0, "Purchased": 1})
x = np.array(data[["Age", "GraduateOrNot",
                    "AnnualIncome", "FamilyMembers",
                    "ChronicDiseases", "FrequentFlyer",
                    "EverTraveledAbroad"]])
y = np.array(data[["TravelInsurance"]])
```

	Age	Employment Type	GraduateOrNot	AnnualIncome	FamilyMembers	ChronicDiseases	FrequentFlyer	EverTravelledAbroad	TravelInsurance
0	31	Government Sector	1	400000	6	1	0	0	0
1	31	Private Sector/Self Employed	1	1250000	7	0	0	0	0
2	34	Private Sector/Self Employed	1	500000	4	1	0	0	1
3	28	Private Sector/Self Employed	1	700000	3	1	0	0	0
4	28	Private Sector/Self Employed	1	700000	8	1	1	0	0
5	25	Private Sector/Self Employed	0	1150000	4	0	0	0	0
6	31	Government Sector	1	1300000	4	0	0	0	0
7	31	Private Sector/Self Employed	1	1350000	3	0	1	1	1
8	28	Private Sector/Self Employed	1	1450000	6	1	1	1	1
9	33	Government Sector	1	800000	3	0	1	0	0

```
In [47]: from sklearn.model_selection import train_test_split
```

35°C Mostly cloudy 17:30 ENG 25-04-2023

4. Uber Trend Analysis for New York City Sales:

Uberanalysis - Jupyter Notebook

localhost:8888/notebooks/Uberanalysis.ipynb

jupyter Uberanalysis Last Checkpoint: a day ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
In [1]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv('uber.csv')
data.head()
```

	Date/Time	Lat	Lon	Base
0	9/1/2014 0:01:00	40.2201	-74.0021	B02512
1	9/1/2014 0:01:00	40.7500	-74.0027	B02512
2	9/1/2014 0:03:00	40.7559	-73.9864	B02512
3	9/1/2014 0:06:00	40.7450	-73.9889	B02512
4	9/1/2014 0:11:00	40.8145	-73.9444	B02512

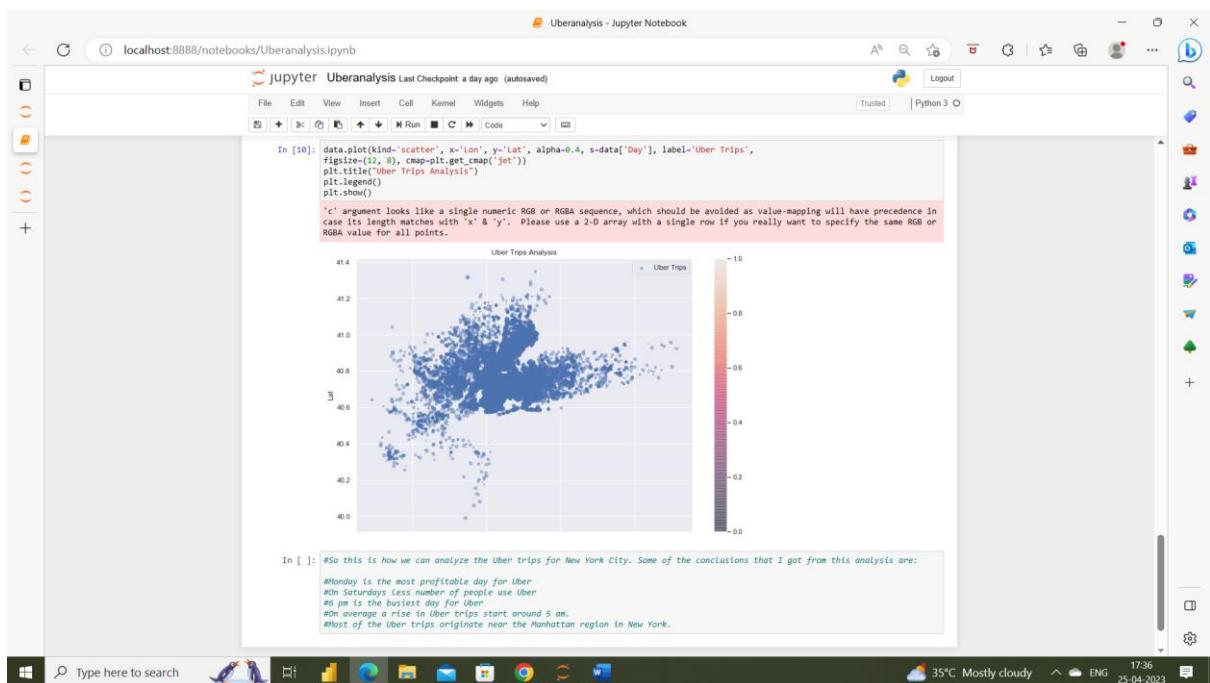
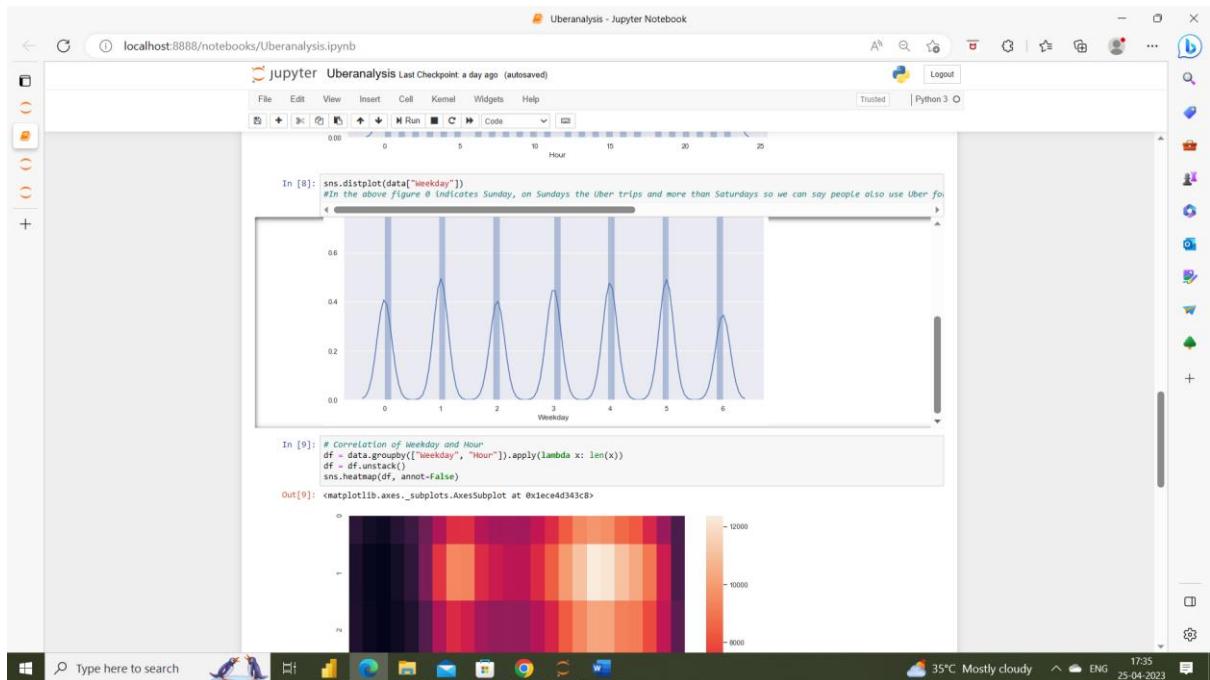
```
In [2]: data["Date/Time"] = data["Date/Time"].map(pd.to_datetime)
```

```
In [3]: data["Day"] = data["Date/Time"].apply(lambda x: x.day)
data["Weekday"] = data["Date/Time"].apply(lambda x: x.weekday())
data["Hour"] = data["Date/Time"].apply(lambda x: x.hour)
```

```
In [4]: data.head(10)
```

	Date/Time	Lat	Lon	Base	Day	Weekday	Hour
0	2014-09-01 00:01:00	40.2201	-74.0021	B02512	1	0	0
1	2014-09-01 00:01:00	40.7500	-74.0027	B02512	1	0	0
2	2014-09-01 00:03:00	40.7559	-73.9864	B02512	1	0	0
3	2014-09-01 00:06:00	40.7450	-73.9889	B02512	1	0	0

35°C Mostly cloudy 17:34 ENG 25-04-2023



5.Water Potability Analysis:

Jupyter bestwaterportability - Jupyter Notebook

localhost:8888/notebooks/bestwaterportability.ipynb

jupyter bestwaterportability Last Checkpoint: 04/13/2023 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [8]:

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

data = pd.read_csv('bestwaterportability.csv')
data.head()
data.shape
```

Out[8]: (3276, 10)

In [5]:

```
data = data.dropna()
data.isnull().sum()
```

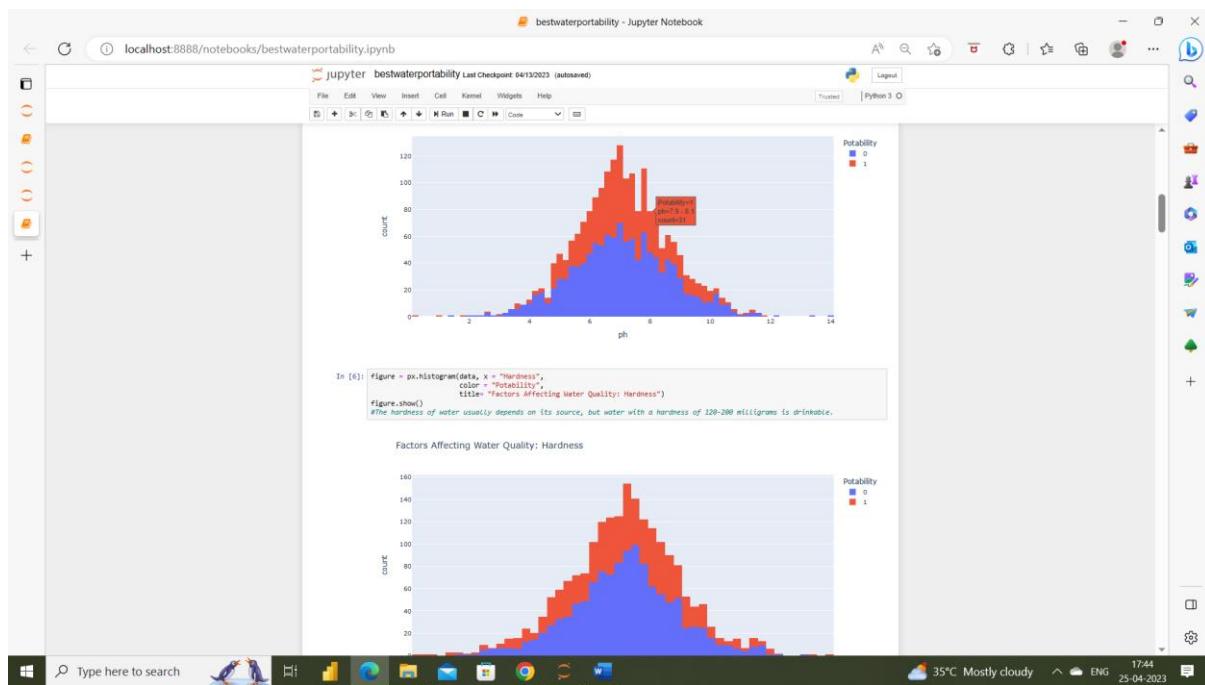
Out[5]:

	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
dtype: int64	0	0	0	0	0	0	0	0	0

In [4]: #The Potability column of this dataset is the column we need to predict because it contains values 0 and 1 that indicate whether

```
plt.figure(figsize=(15, 10))
sns.countplot(data.Potability)
plt.title("Distribution of Unsafe and Safe Water")
plt.show()
```

35°C Mostly cloudy 1742 25-04-2023



6.IRIS Classification Analysis

Jupyter irisdatapredictionclassification - Jupyter Notebook

In [2]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
iris = pd.read_csv("irisdata.csv")
```

In [3]: iris.head(10)

Out[3]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa

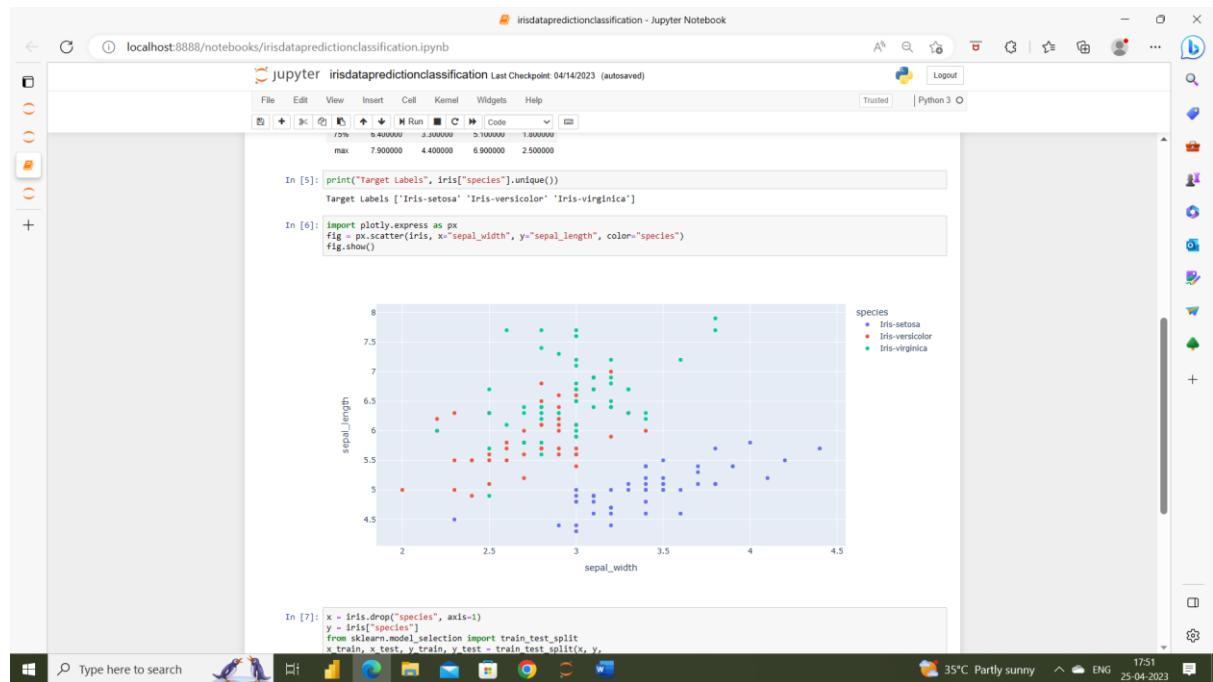
In [4]: iris.describe()

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.826096	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.000000	2.800000	1.600000	0.300000
50%	5.000000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [5]: print("Target Labels", iris["species"].unique())

Out[5]: Target Labels ['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']



7.Sentiment Analysis eg. Flipcart

SentimentAnalysisFlipcart - Jupyter Notebook

localhost:8888/notebooks/SentimentAnalysisFlipcart.ipynb

Jupyter SentimentAnalysisFlipcart Last Checkpoint: 04/14/2023 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [3]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from nltk.sentiment.vader import SentimentIntensityAnalyzer
data = pd.read_csv('flipkartfile.csv')
print(data.head())
Number          Product_name \
0      0  Lenovo Ideapad Gaming 3 Ryzen 5 Hexa Core 5600...
1      1  Lenovo Ideapad Gaming 3 Ryzen 5 Hexa Core 5600...
2      2  Lenovo Ideapad Gaming 3 Ryzen 5 Hexa Core 5600...
3      3  DELL Inspiron 15 3508 - (4 GB/1...
4      4  DELL Inspiron Athlon Dual Core 3050U - (4 GB/1...
                                                 Review Rating
0  Best under 60k Great performance got it for a...      5
1  Good performance but usually it has also that...      5
2  Great performance but usually it has also that...      5
3  My wife is so happy and best product 😊😊      5
4  Light weight laptop with new amazing features,...      5
```

In [4]:

```
data.shape
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
```

In [8]:

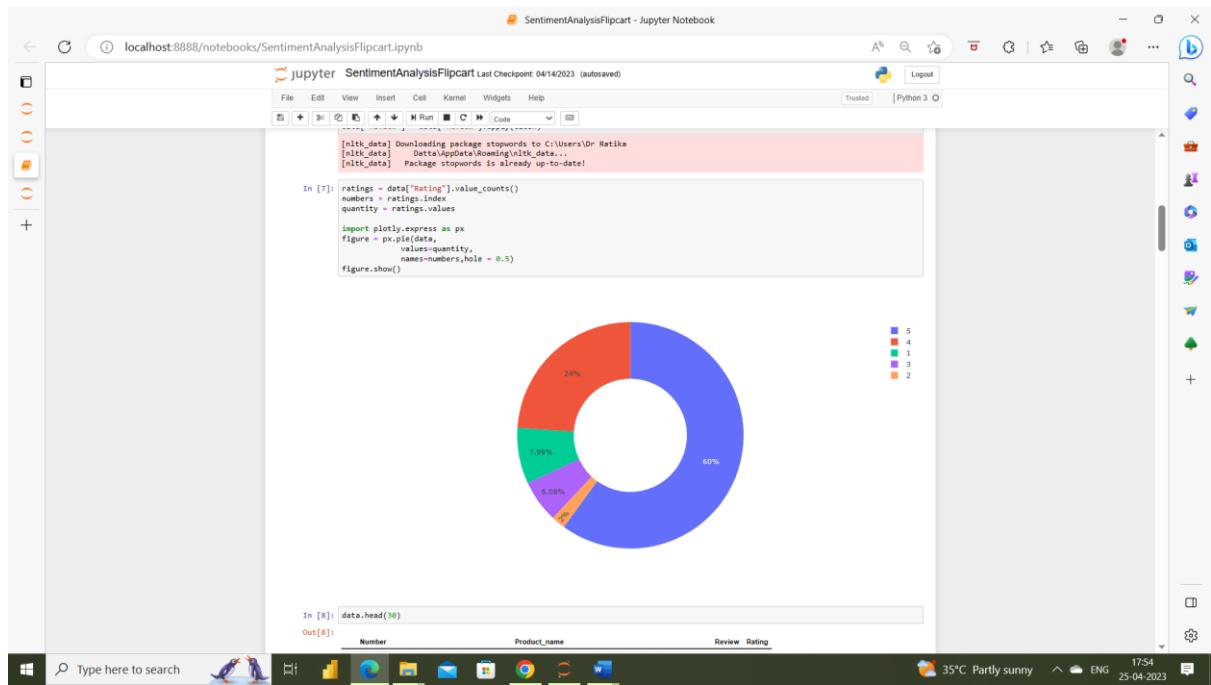
```
data.info()
```

In [5]:

```
print(data.isnull().sum())
```

Number 0
Product_name 0
Review 0
Rating 0

35°C Partly sunny 1752 25-04-2023



The screenshot shows a Jupyter Notebook interface with the title "SentimentAnalysisFlipcart - Jupyter Notebook". The notebook contains Python code for sentiment analysis and a word cloud visualization.

In [10]:

```
text = " ".join([r for r in data.Review])
stopwords = set(STOPWORDS)
wordcloud = WordCloud(stopwords=stopwords,
                      background_color="white").generate(text)
plt.figure(figsize=(10,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

In [11]:

```
nltk.download('vader_lexicon')
sentiments = SentimentIntensityAnalyzer()
data['Positive'] = [sentiments.polarity_scores(r["pos"]) for r in data["Review"]]
data['Negative'] = [sentiments.polarity_scores(r["neg"]) for r in data["Review"]]
data['Neutral'] = [sentiments.polarity_scores(r["neutral"]) for r in data["Review"]]
data[['Review', 'Positive', 'Negative', 'Neutral']]
print(data.head())
[nltk_data] Downloading package vader_lexicon to C:\Users\Dr Ratika
[nltk_data]   Data-0p0taRoaming\nltk_data...
[nltk_data] Data-0p0taRoaming\nltk_data...
[nltk_data] Package vader_lexicon is already up-to-date!
```

Review Positive Negative Neutral

35°C Partly sunny ENG 17:45 25-04-2023

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** SentimentAnalysisFlipkart - Jupyter Notebook
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Logout
- Code Cells:**
 - In [2]: `data.tail(10)`
 - Out[2]:

```
NameError: name 'data' is not defined
```
 - In [71]: `data.head(10)`
 - Out[71]:

Number	Product_name	Review	Rating	Analysis
0	Lenovo Ideapad Gaming 3 Ryzen 5 Hexa Core 5600.	Best under 60k Great performance got it for a...	5	Positive
1	Lenovo Ideapad Gaming 3 Ryzen 5 Hexa Core 5600.	Good performance...	5	Positive
2	Lenovo Ideapad Gaming 3 Ryzen 5 Hexa Core 5600.	Great performance but usually it has also that...	5	Positive
3	DELL Inspiron Athon Dual Core 3500 - i4 GB@2...	My wife is so happy and best product 😊	5	Positive
4	DELL Inspiron Athon Dual Core 3500 - i4 GB@2...	Light weight laptop with new amazing features...	5	Positive
5	DELL Inspiron Athon Dual Core 3500 - i4 GB@2...	Amazing laptop. am so much happy. thanks for F...	5	Positive
6	DELL Inspiron Athon Dual Core 3500 - i4 GB@2...	Over all a good laptop for personal use	5	Positive
7	DELL Inspiron Athon Dual Core 3500 - i4 GB@2...	Thank you so much Flipkart	4	Positive
8	DELL Inspiron Athon Dual Core 3500 - i4 GB@2...	Amazing product	5	Positive
9	DELL Inspiron Athon Dual Core 3500 - i4 GB@2...	Good for normal work, students, online classe...	3	Neutral
 - In [81]:

```
NameError: name 'data' is not defined
```
 - In [17]:

```
1 positiveReviews = data.loc[data['Analysis']=='Positive']
```
 - Out[17]:

Review	Positive	Negative	Neutral	
0	best great performance got around backup bi...	0.395	0.101	0.504

8.Language Prediction Analysis:

Jupyter Notebook - languagedetection

```
In [5]: import pandas as pd
import numpy as np
from sklearn.feature_selection import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB

In [6]: data = pd.read_csv("https://raw.githubusercontent.com/amankharwal/Website-data/master/dataset.csv")

In [7]: data.head(10)

Out[7]:
Text language
0 klement gotwaldi sumukhe patsameeril ning ... Estonian
1 sebes joseph perera thomas på eng the jesuit... Swedish
2 osusdagis émertől thanon charon knung ... Thai
3 வினாவுடைய குறிப்புகள் மூலம் உபதிட... Tamil
4 de spons behoorlt het geslaach hallocone en... Dutch
5 ジノイドが行きかたりバソに轟つてしまい、彼分が轟くなつて聞こむわが、今すぐさを隠さないにて... Japanese
6 Institut implore toutes les canadienne abet... Turkish
7 muller mos figura centralis circulum doctora... Latin
8 ملکہ الکریمہ علیہ السلام میرزا محمد علی... Urdu
9 シーニー・ワールドは、アン・ベルナルド・アベニュー沿い西セント・ポーリング高校に... Japanese

In [8]: data.isnull().sum()

Out[8]:
Text    0
language    0
dtype: int64

In [9]: data["language"].value_counts()

Out[9]:
Indonesian    1000
Tamil    1000
Dutch    1000
Hindi    1000
Arabic    1000
Romanian    1000
Estonian    1000
Swedish    1000
Spanish    1000
Latin    1000
Korean    1000
Persian    1000
Portuguese    1000
Turkish    1000
English    1000
Japanese    1000
Pushko    1000
Urdu    1000
Thai    1000
Chinese    1000
Russian    1000
French    1000
Name: language, dtype: int64
```

Type here to search 35°C Partly sunny 17:58 25-04-2023

Jupyter Notebook - languagedetection

```
In [10]: x = np.array(data["Text"])
y = np.array(data["language"])

cv = CountVectorizer()
X = cv.fit_transform(x)
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.33,
                                                    random_state=42)

In [11]: model = MultinomialNB()
model.fit(X_train,y_train)
model.score(X_test,y_test)

Out[11]: 0.953168044077135

In [12]: user = input("Enter a Text: ")
data = cv.transform([user]).toarray()
output = model.predict(data)
print(output)

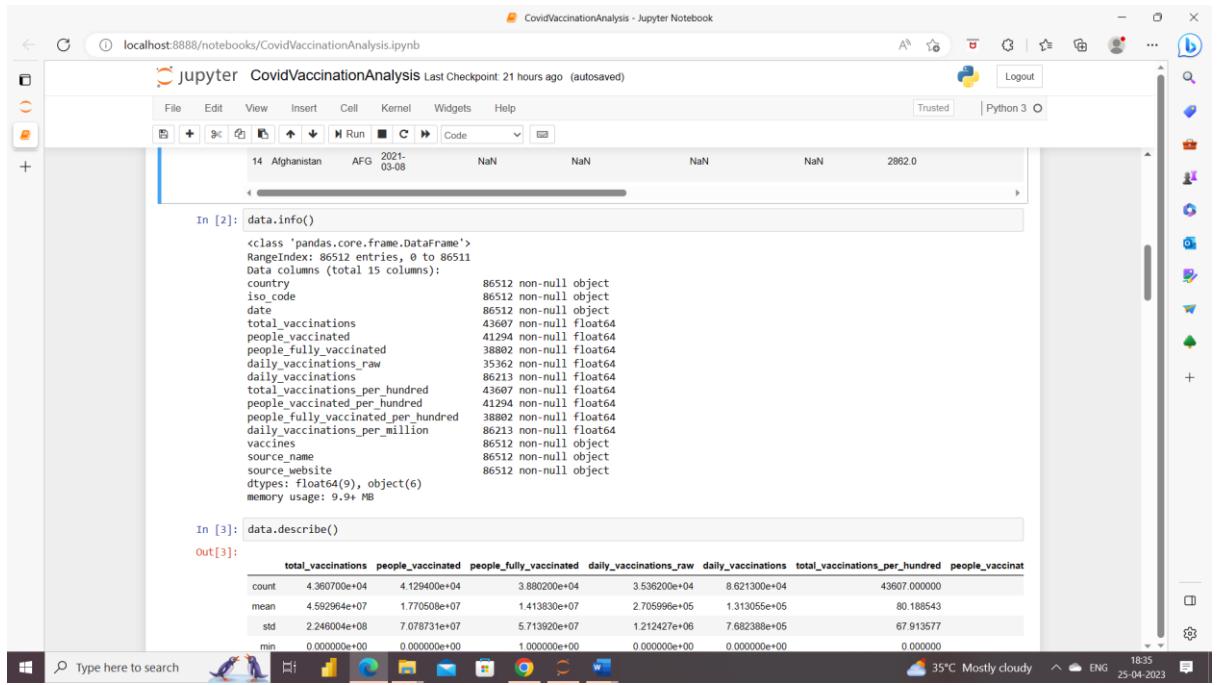
Enter a Text: i love genuinity
['English']

In [13]: user = input("Enter a Text: ")
data = cv.transform([user]).toarray()
output = model.predict(data)
print(output)

Enter a Text: comprenant le plus souvent la surréalisme et les ganglions situés à proximité
['French']
```

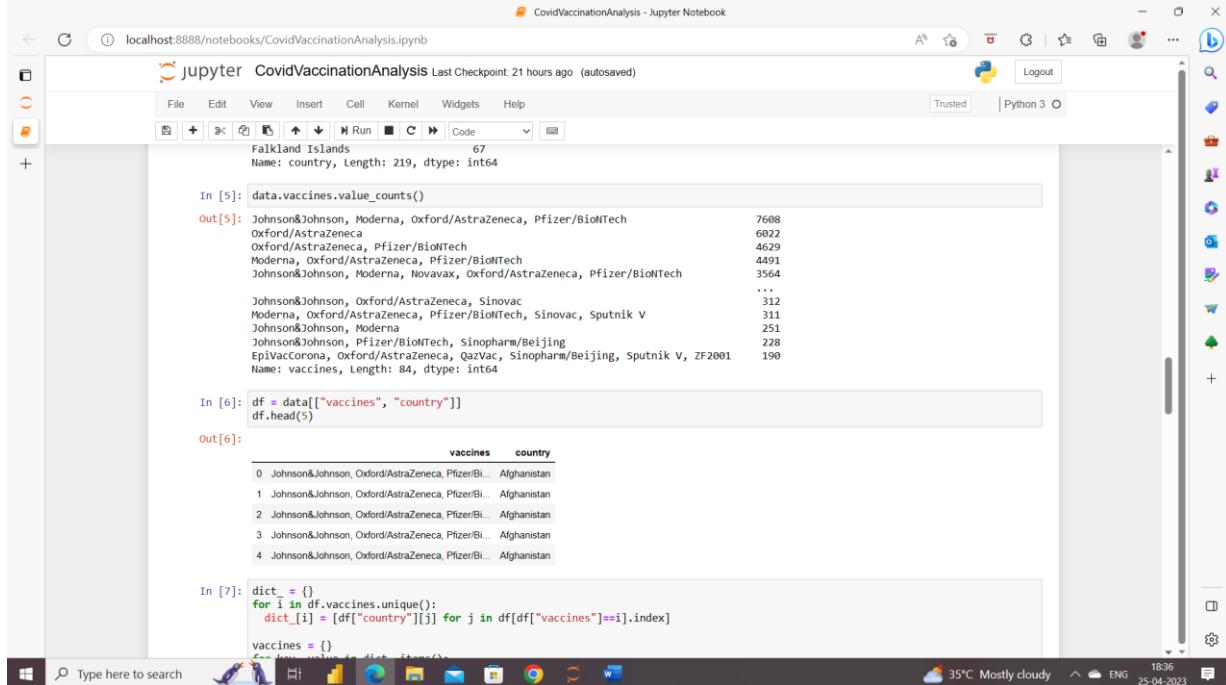
Type here to search 35°C Partly sunny 17:58 25-04-2023

9. Covid19 Vaccine Analysis



```
In [2]: data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 86512 entries, 0 to 86511
Data columns (total 15 columns):
country                           86512 non-null object
iso_code                           86512 non-null object
date                               86512 non-null object
total_vaccinations                 86512 non-null float64
people_vaccinated                  41294 non-null float64
people_fully_vaccinated            38802 non-null float64
daily_vaccinations_raw             35362 non-null float64
daily_vaccinations                86213 non-null float64
total_vaccinations_per_hundred    43607 non-null float64
people_vaccinated_per_hundred     41294 non-null float64
people_fully_vaccinated_per_hundred 38802 non-null float64
daily_vaccinations_per_million    86213 non-null float64
vaccined                           86512 non-null object
source_name                         86512 non-null object
source_website                      86512 non-null object
dtypes: float64(9), object(6)
memory usage: 9.9+ MB

In [3]: data.describe()
Out[3]:
   total_vaccinations  people_vaccinated  people_fully_vaccinated  daily_vaccinations_raw  daily_vaccinations  total_vaccinations_per_hundred  people_vaccinat
count      4.360700e+04        4.129400e+04        3.880200e+04        3.536200e+04        8.621300e+04        4.3607000000
mean       4.592964e+07        1.770508e+07        1.413830e+07        2.705996e+05        1.310556e+05        80.188543
std        2.246004e+08        7.078731e+07        5.713920e+07        1.214247e+06        7.682388e+05        67.913577
min        0.000000e+00        0.000000e+00        1.000000e+00        0.000000e+00        0.000000e+00        0.000000
max       1.000000e+09        1.000000e+09        1.000000e+09        1.000000e+09        1.000000e+09        1.000000e+09
```



```
Falkland Islands          67
Name: country, length: 219, dtype: int64

In [5]: data.vaccines.value_counts()
Out[5]:
Johnson&Johnson, Moderna, Oxford/AstraZeneca, Pfizer/BioNTech    7608
Oxford/AstraZeneca                                                 6022
Oxford/AstraZeneca, Pfizer/BioNTech                                4629
Moderna, Oxford/AstraZeneca, Pfizer/BioNTech                        4491
Johnson&Johnson, Moderna, Novavax, Oxford/AstraZeneca, Pfizer/BioNTech 3564
...
Johnson&Johnson, Oxford/AstraZeneca, Sinovac                         312
Moderna, Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac, Sputnik V    311
Johnson&Johnson, Moderna                                         251
Johnson&Johnson, Pfizer/BioNTech, Sinopharm/Beijing                   228
EpivacCorona, Oxford/AstraZeneca, QazVac, Sinopharm/Beijing, Sputnik V 190
Name: vaccines, Length: 84, dtype: int64

In [6]: df = data[["vaccines", "country"]]
df.head(5)
Out[6]:
   vaccines      country
0  Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac, Sputnik V, EpivacCorona, QazVac, Sinopharm/Beijing, Sputnik V, ZF2001  Afghanistan
1  Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac, Sputnik V, EpivacCorona, QazVac, Sinopharm/Beijing, Sputnik V, ZF2001  Afghanistan
2  Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac, Sputnik V, EpivacCorona, QazVac, Sinopharm/Beijing, Sputnik V, ZF2001  Afghanistan
3  Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac, Sputnik V, EpivacCorona, QazVac, Sinopharm/Beijing, Sputnik V, ZF2001  Afghanistan
4  Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac, Sputnik V, EpivacCorona, QazVac, Sinopharm/Beijing, Sputnik V, ZF2001  Afghanistan

In [7]: dict_ = {}
for i in df.vaccines.unique():
    dict_[i] = [df[df["country"] == i].index]
    vaccines = []
    for index in dict_[i]:
        vaccines.append(df.loc[index])
```

CovidVaccinationAnalysis - Jupyter Notebook

localhost:8888/notebooks/CovidVaccinationAnalysis.ipynb

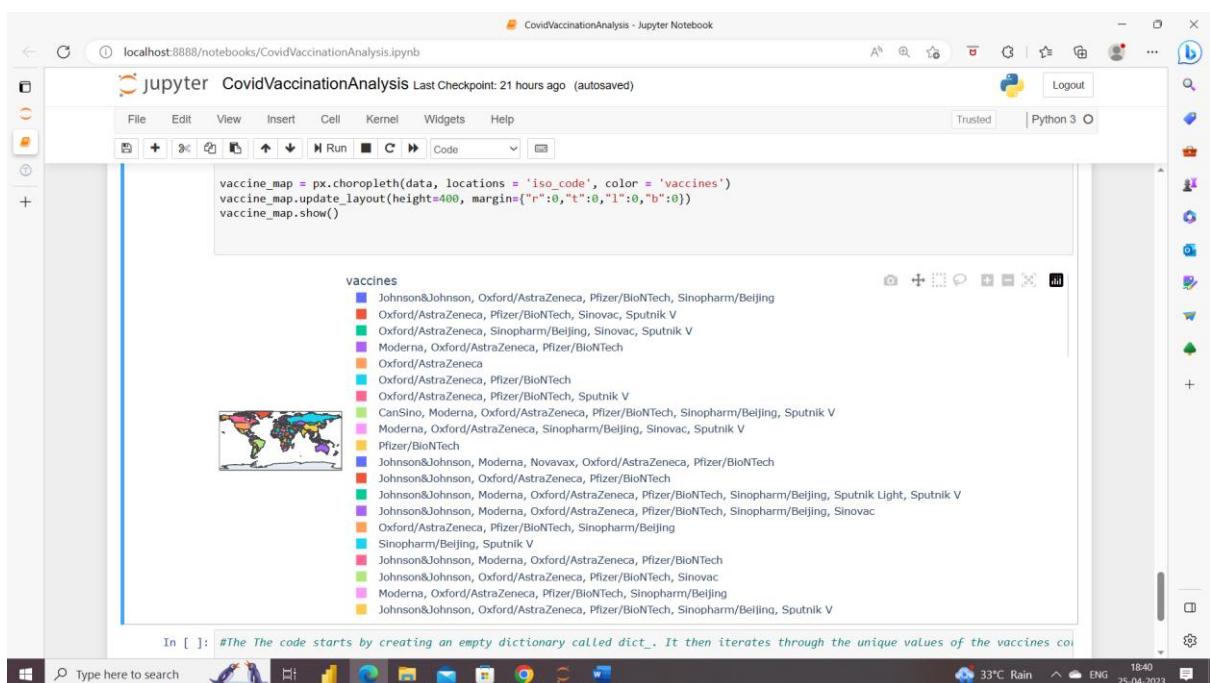
jupyter CovidVaccinationAnalysis Last Checkpoint: 21 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
In [7]: dict_ = {}
for i in df.vaccines.unique():
    dict_[i] = df["country"][i]
for j in df[df["vaccines"]==i].index:
    vaccines = []
    for key, value in dict_.items():
        vaccines.append(value)
    for i, j in vaccines.items():
        print(f'{i}:::{j}')
```

Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing::Belize, 'Namibia', 'Trinidad and Tobago', 'Afghanistan', 'Cameroun'
 Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac, Sputnik V::Albania, 'Azerbaijan', 'Bosnia and Herzegovina', 'Oman'
 Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac, Sputnik V::Zimbabwe, 'Algeria'
 Moderna, Oxford/AstraZeneca, Pfizer/BioNTech::United Kingdom, 'Isle of Man', 'Japan', 'Sint Maarten (Dutch part)', 'Fiji',
 'Guernsey', 'Jersey', 'Sweden', 'Andorra', 'Malta', 'Pitcairn', 'Falkland Islands', 'Liberia', 'Sao Tome and Principe', 'Tuv
 ala', 'Vanuatu', 'Angola', 'Samoa', 'Tonga', 'Solomon Islands', 'Timor', 'Vanuatu', 'Saint Helena', 'Papua New Guinea', 'Democ
 ratic Republic of Congo', 'Togo', 'Saint Lucia', and the (predominant)
 Oxford/AstraZeneca, Pfizer/BioNTech::'Nigeria', 'Monaco', 'Mali', 'Antigua and Barbuda',
 CanSino, Moderna, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sputnik V::Argentina'
 Moderna, Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac, Sputnik V::Armenia'
 Pfizer/BioNTech::'Cook Islands', 'Niue', 'Tokelau', 'Aruba', 'Turks and Caicos Islands', 'Monaco', 'New Caledonia'
 Johnson&Johnson, Moderna, Novavax, Oxford/AstraZeneca, Pfizer/BioNTech::'Netherlands', 'South Korea', 'Italy', 'Czechia', 'Ge
 rmany', 'Slovenia', 'Austria', 'Lithuania'
 Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioNTech::Eswatini, 'Grenada', 'Bahamas'
 Johnson&Johnson, Moderna, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sputnik Light, Sputnik V::Bahrain'
 Johnson&Johnson, Moderna, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sinovac::Bangladesh'
 Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing::Barbados, 'Maldives', 'Suriname', 'Peru', 'Dominica'
 Sinopharm/Beijing, Sputnik V::Belarus', 'Kyrgyzstan'
 Johnson&Johnson, Moderna, Oxford/AstraZeneca, Pfizer/BioNTech::Romania', 'Croatia', 'Jamaica', 'Luxembourg', 'Portugal', 'Gr
 eece', 'Spain', 'Cyprus', 'Canada', 'Belgium', 'Bulgaria', 'Estonia', 'Ireland', 'France', 'Malta', 'Poland')

35°C Mostly cloudy 1837 25-04-2023



10. Electricity Price Prediction:

Jupyter Electricitypricepredictionpython - Jupyter Notebook

In [2]:

```
import pandas as pd
import numpy as np
data = pd.read_csv('Electricitypricepred.csv')
print(data.head())
```

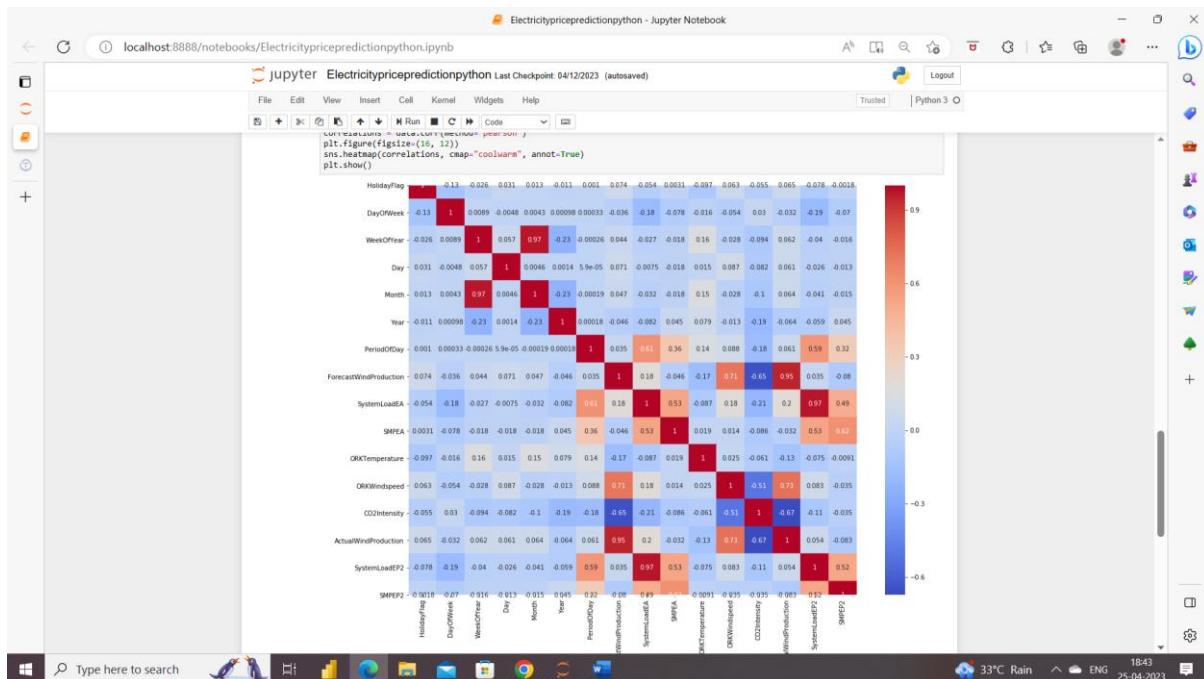
	DateTime	Holiday	HolidayFlag	DayOfWeek	WeekOfYear	Day	Month	Year	PeriodOfDay	ForecastWindProduction	SystemLoadEA	SMPEA
0	01-11-2011 00:00	None	0	1	44	1	11	2011	0	315.31	3388.77	49.26
1	01-11-2011 00:30	None	0	1	44	1	11	2011	1	321.8	3196.66	49.26
2	01-11-2011 01:00	None	0	1	44	1	11	2011	2	328.57	3060.71	49.1
3	01-11-2011 01:30	None	0	1	44	1	11	2011	3	335.6	2945.56	48.04
4	01-11-2011 02:00	None	0	1	44	1	11	2011	4	342.9	2849.34	33.75

ORKTemperature ORKWindSpeed CO2Intensity ActualWindProduction SystemLoadEP2 \

	0	1	2	3	4
0	6	11.1	11.1	9.3	11.1
1	600.71	605.42	589.97	585.94	571.52
2	356	317	311	313	346
3	3159.6	2973.01	2834	2725.99	2655.64
4					

SMPEP2

	0	1	2	3	4
0	54.32	54.23	54.23	53.47	
1					
2					
3					
4					



Electricitypricepredictionpython - Jupyter Notebook

In [10]:

```
x = data[["Day", "Month", "ForecastWindProduction", "SystemLoadEA", "SMPEA", "ORKTemperature", "ORKWindSpeed", "CO2Intensity", "ActualWindProduction", "SystemLoadEP2"]]
y = data["SMPEP"]
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2, random_state=42)
```

In [11]:

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor()
model.fit(xtrain, ytrain)

D:\python\lib\site-packages\sklearn\ensemble\forest.py:245: FutureWarning: The default value of n_estimators will change from 10 in version 0.20 to 100 in 0.22. FutureWarning
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[11]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=10,
                      n_jobs=None, oob_score=False, random_state=None,
                      verbose=0, warm_start=False)
```

In [12]:

```
#features = [[{"Day": 10, "Month": 12, "ForecastWindProduction": 54.1, "SystemLoadEA": 4241.05, "SMPEA": 40.56, "ORKTemperature": 14.8, "ORKWindSpeed": 491.32, "CO2Intensity": 54.0, "ActualWindProduction": 4426.84}]]
model.predict(features)
```

Out[12]:

```
array([67.289])
```

In [1]: #Predicting the price of electricity helps a lot of companies to understand how much electricity expenses they have to pay every month

11. Bank Churn Analysis

Bank Churn Trend analysis - Jupyter Notebook

In [1]:

```
import pandas as pd
```

In [2]:

```
import numpy as np
```

In [4]:

```
data=pd.read_csv('churn in banking.csv')
```

In [5]:

```
data.head(10)
```

Out[5]:

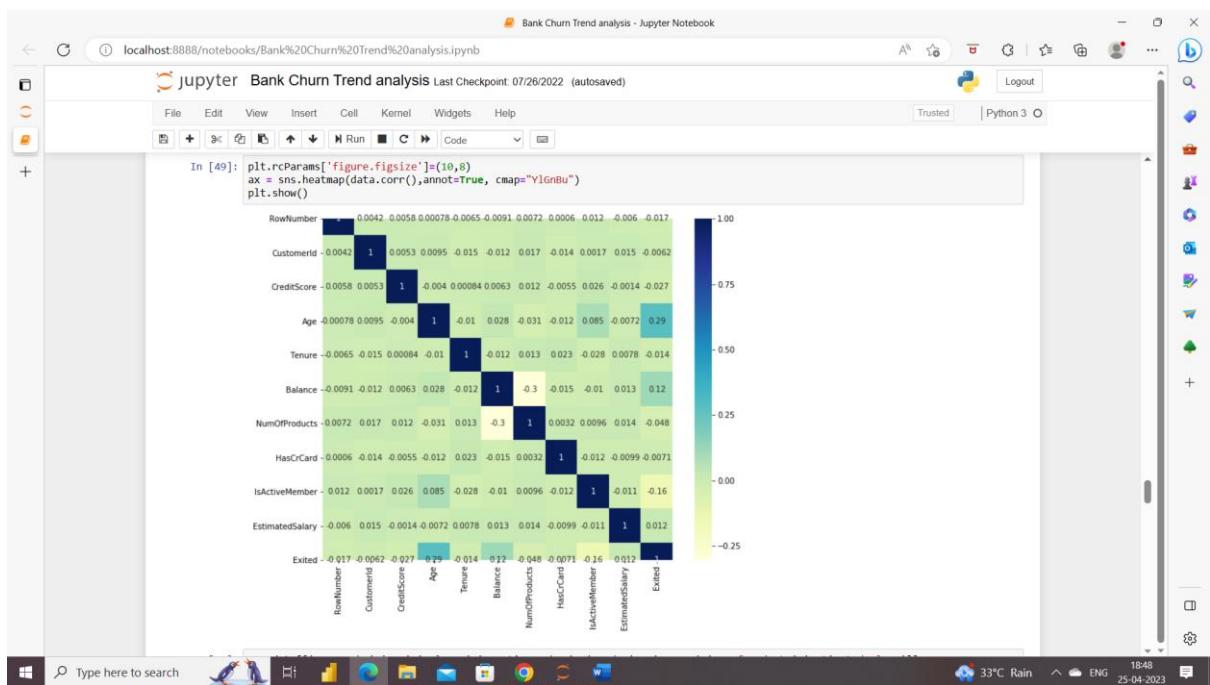
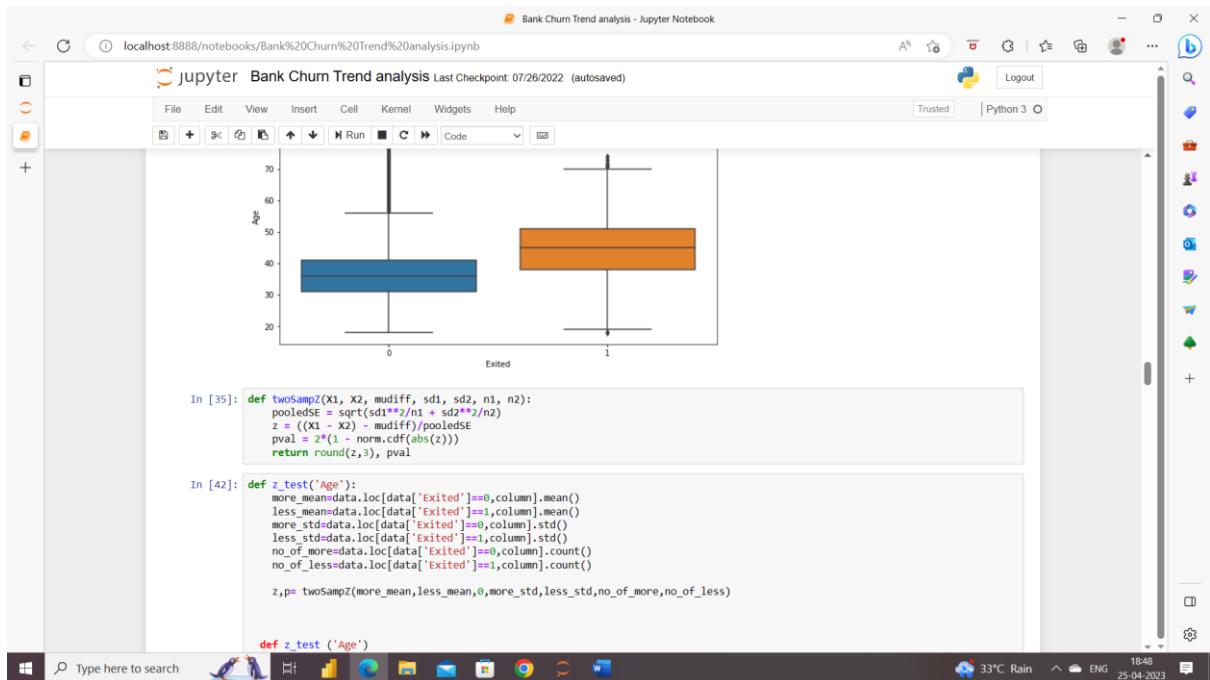
RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	1	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88
1	2	15647311	Hill	608	Spain	Female	41	1 83807.86	1	0	1	112542.58
2	3	15619304	Onio	502	France	Female	42	8 159860.80	3	1	0	113931.57
3	4	15701354	Boni	699	France	Female	39	1 0.00	2	0	0	93826.63
4	5	15737888	Mitchell	850	Spain	Female	43	2 125510.82	1	1	1	79084.10
5	6	15574012	Chu	645	Spain	Male	44	8 113755.78	2	1	0	149756.71
6	7	15592531	Bartlett	822	France	Male	50	7 0.00	2	1	1	10062.80
7	8	15656148	Obinna	376	Germany	Female	29	4 115046.74	4	1	0	119346.88
8	9	15792365	He	501	France	Male	44	4 142051.07	2	0	1	74940.50
9	10	15592389	H?	684	France	Male	27	2 134603.88	1	1	1	71725.73

In [6]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
RowNumber      10000 non-null int64
CustomerId     10000 non-null int64

```



12. Google Word Trend Analysis

Googlesearchanalysis - Jupyter Notebook

localhost:8888/notebooks/Googlesearchanalysis.ipynb

jupyter Googlesearchanalysis Last Checkpoint: Yesterday at 5:09 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [1]: `!pip install pytrends`

```
Collecting pytrends
  Downloading https://files.pythonhosted.org/packages/08/ba/7a243723c790000fafb8056ff1cc46fd129f46d0d353837938a670c4d23/pytrends-4.9.2-py3-none-any.whl
Requirement already satisfied: requests>=2.0 in d:\python\lib\site-packages (from pytrends) (2.22.0)
Requirement already satisfied: pandas>=0.25 in d:\python\lib\site-packages (from pytrends) (0.25.1)
Requirement already satisfied: lxml in d:\python\lib\site-packages (from pytrends) (4.4.1)
Requirement already satisfied: idna<2.9,*>_<2.5 in d:\python\lib\site-packages (from requests>=2.0->pytrends) (2.8)
Requirement already satisfied: urllib3!=1.25.0,>_<1.26,>_>1.21.1 in d:\python\lib\site-packages (from requests>=2.0->pytrends) (1.24.2)
Requirement already satisfied: charsetencodings<3.1.0,>_>0.2 in d:\python\lib\site-packages (from requests>=2.0->pytrends) (3.0.4)
Requirement already satisfied: certifi>2017.4.17 in d:\python\lib\site-packages (from requests>=2.0->pytrends) (2019.9.11)
Requirement already satisfied: python-dateutil>=2.6.1 in d:\python\lib\site-packages (from pandas>=0.25->pytrends) (2.8.0)
Requirement already satisfied: pytz>=2017.2 in d:\python\lib\site-packages (from pandas>=0.25->pytrends) (2019.3)
Requirement already satisfied: numpy>=1.13.3 in d:\python\lib\site-packages (from pandas>=0.25->pytrends) (1.16.5)
Requirement already satisfied: six>=1.5 in d:\python\lib\site-packages (from python-dateutil>=2.6.1->pandas>=0.25->pytrends) (1.12.0)
Installing collected packages: pytrends
Successfully installed pytrends-4.9.2
```

In [2]: `import pandas as pd
from pytrends.request import TrendReq
import matplotlib.pyplot as plt
trends = TrendReq()`

In [5]: `trends.build_payload(kw_list=["Machine Learning"])
data = trends.interest_by_region()
data = data.sort_values(by="Machine Learning", ascending=False)
data = data.head(10)
print(data)`

Machine Learning

Type here to search

geoName
China

Machine Learning

33°C Rain ENG 25-04-2023 1851

