COVID RISK PREDICTION

Team Members : Balaji Sai Charan, Jalukuru(CWID – 885177295)
Nikitha Reddy ,Margadi(CWID – 885177147)

INTRODUCTION –

Most people infected with COVID-19 virus will recover without requiring special treatment but sometimes may experience mild respiratory illness. Older people, and those with underlying medical problems are more likely to develop serious illness. Shortage of medical resources and an effective plan to disperse them has been one of the main issues that healthcare workers have encountered throughout the pandemic. The main goal of this project is to build a predictive model with given Covid-19 patient's current symptom and medical history will predict whether the patient is in high risk or not using our model and also analyze data to draw some conclusions based on data available using Google Cloud Platform, Spark and Streamlit.

PROJECT SCOPE –

- Main objective of this project aims to predict whether the patients with COVID-19 and with any other past medical history are at high risk of death or not.

- We will also analyze data by comparing various columns like
  - Immune system vs DEATHS across Age groups.
  - Distribution of Covid cases across all age groups and among Gender.
  - Which Diseases are more prominent among Males and Females.

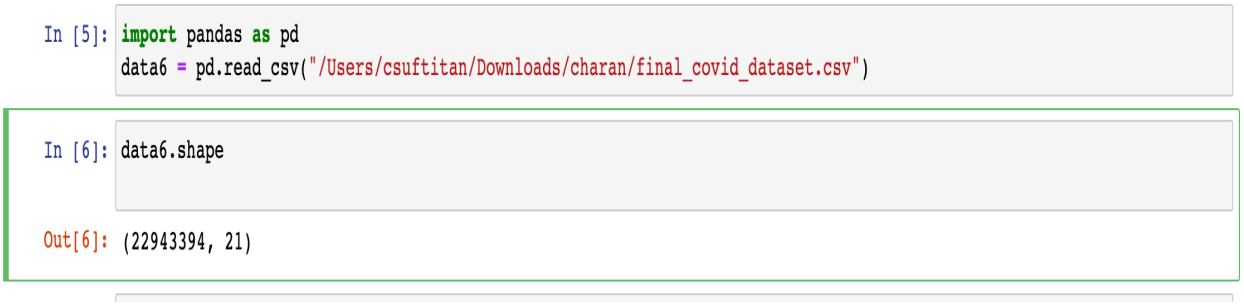
TECHNOLOGIES USED-

- Data Ingestion : SQOOP,GCS, BigQuery (relational Database)
- Data Analysis : pyspark, python3
- Data Prediction : pyspark,python3
- Libraries Used : matplotlib, seaborn, pandas, numpy ,spark libs
- Cluster : Data proc(1 master , 2 worker nodes)
- Ide : jupyter notebook.

FUNCTIONALITIES –

1. DATASET:

   We have downloaded our dataset from the Kaggle Data Sets.Our Dataset consists of 22943394 rows and 21 columns of Data. It is a data set which consists of 2GB of data.



2. Our Data Set contains 21 columns named below:

```
In [34]: data6.columns

Out[34]: Index(['USMER', 'MEDICAL_UNIT', 'SEX', 'PATIENT_TYPE', 'PNEUMONIA', 'AGE',
                'PREGNANT', 'DIABETES', 'COPD', 'ASTHMA', 'INMSUPR', 'HIPERTENSION',
                'OTHER_DISEASE', 'CARDIOVASCULAR', 'OBESITY', 'RENAL_CHRONIC',
                'TOBACCO', 'CLASIFFICATION_FINAL', 'DEATH'],
               dtype='object')
```

- sex: 1 for female and 2 for male.
- age: of the patient.
- classification: covid test findings. Values 1-3 mean that the patient was diagnosed with covid in different degrees. 4 or higher means that the patient is not a carrier of covid or that the test is inconclusive.
- patient type: type of care the patient received in the unit. 1 for returned home and 2 for hospitalization.
- pneumonia: whether the patient already have air sacs inflammation or not.
- pregnancy: whether the patient is pregnant or not.
- diabetes: whether the patient has diabetes or not.
- copd: Indicates whether the patient has Chronic obstructive pulmonary disease or not.
- asthma: whether the patient has asthma or not.
- inmsupr: whether the patient is immunosuppressed or not.
- hypertension: whether the patient has hypertension or not.
- cardiovascular: whether the patient has heart or blood vessels related disease.
- renal chronic: whether the patient has chronic renal disease or not.
- other disease: whether the patient has other disease or not.
- obesity: whether the patient is obese or not.
- tobacco: whether the patient is a tobacco user.
- usmr: Indicates whether the patient treated medical units of the first, second or third level.
- medical unit: type of institution of the National Health System that provided the care.
- intubed: whether the patient was connected to the ventilator.

- icu: Indicates whether the patient had been admitted to an Intensive Care Unit.
- date died: If the patient died indicate the date of death, and 9999-99-99 otherwise.

3. Google cloud Platform:

   we can manage our Google Cloud projects and resources using the web-based, graphical user interface provided by the Google Cloud console. When using the Google Cloud console, you first select an existing project or start a new one before using the resources you create within that project.

   We used a lot of tools in GCP to complete our project.

   - **GCS (google cloud storage):** Any objects can be stored in Google Cloud using the service known as cloud storage. A file in any format that makes up an object is an immutable piece of data.
   - **BigQuery:** BigQuery is a fully managed enterprise data warehouse that aids in managing and analyzing your data.
   - **DataProc:** You can use open source data tools for batch processing, querying, streaming, and machine learning through Dataproc, a managed Spark and Hadoop service.
   - **AI Platform:** AI Platform is a managed service that enables you to easily build and Deploy machine learning models, that work on any type of data, of any size.

4. Data Cleaning:

   Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset there are many opportunities for data to be duplicated or mislabeled.

We used Pandas DataFrame for cleaning initially the data will be in Spark Dataframe we converted it to Pandas and Continued out cleaning

- Remove duplicate or irrelevant observations:

  Remove unwanted observations from your dataset, including duplicate observations or irrelevant observations.so we removed duplicates from our data set.

- Handle missing data:

  You can’t ignore missing data because many algorithms will not accept missing values. There are a couple of ways to deal with missing data. Neither is optimal, but both can be considered. One is to remove null values and second is to fill null values by mean , median or mode. In our data set features like
  Dropped “Intubed", "Pregnant" and "Icu” columns which have more than 50% of null values. And features like
  “Pneumonia", "Diabetes", "Copd", "Asthma", "Insumpr", "Hipertension", "Other_diseases", "Cardiovascular", "Obesity", "Renal_chronic" and "Tobacco" have a low null percentage so we used them for training.

- Validate and QA-

  At the end of the data cleaning process, you should be able to answer these questions as a part of basic validation .
  Does the data make sense? So to validate this statement we changes couple of values to make the data more understandable
  As their "2" values means "0" (Negative Boolean), we transformed 2's into 0's, and 97, 98 and 99 values into nan and also changed "DATE_DIED" column with understandable entries like 0 if the person is alive (9999-99-99 value), and 0 if not (value != 9999-99-99). We'll also change its name to "DIED".

5. Data Analysis:

   Data analysis is the process of modifying, processing, and cleaning raw data in order to extract useful, pertinent information that supports decision-making. The process offers helpful insights and statistics, frequently presented in charts, images, tables, and graphs, which reduce the risks associated with decision-making.

   We Used BigQuery, DataProc and Spark for this task.
   We have Done Following Analysis in our data :

- A plot showing us the distribution of cases over age.
- Which age of Patients we sent home mostly after Medication.
- Prominent diseases among males and females that caused death.
- No of deaths age wise among males and females
- Percentage of people around 12 to 30 having weak immunity
- Percentage of people around 30 to 100 having weak immunity

6. Data Prediction :

- The process of using data analytics to make predictions based on data.
- We Used Google Cloud Platform for Our project to do this task we used Dataproc and Pyspark with AI Platform.
- We experimented a long list of Machine learning algorithms including Random Forests, Decision Trees , Naïve Bayes etc
- Only by Using Logistic Regression we were able to achieve higher accuracy.

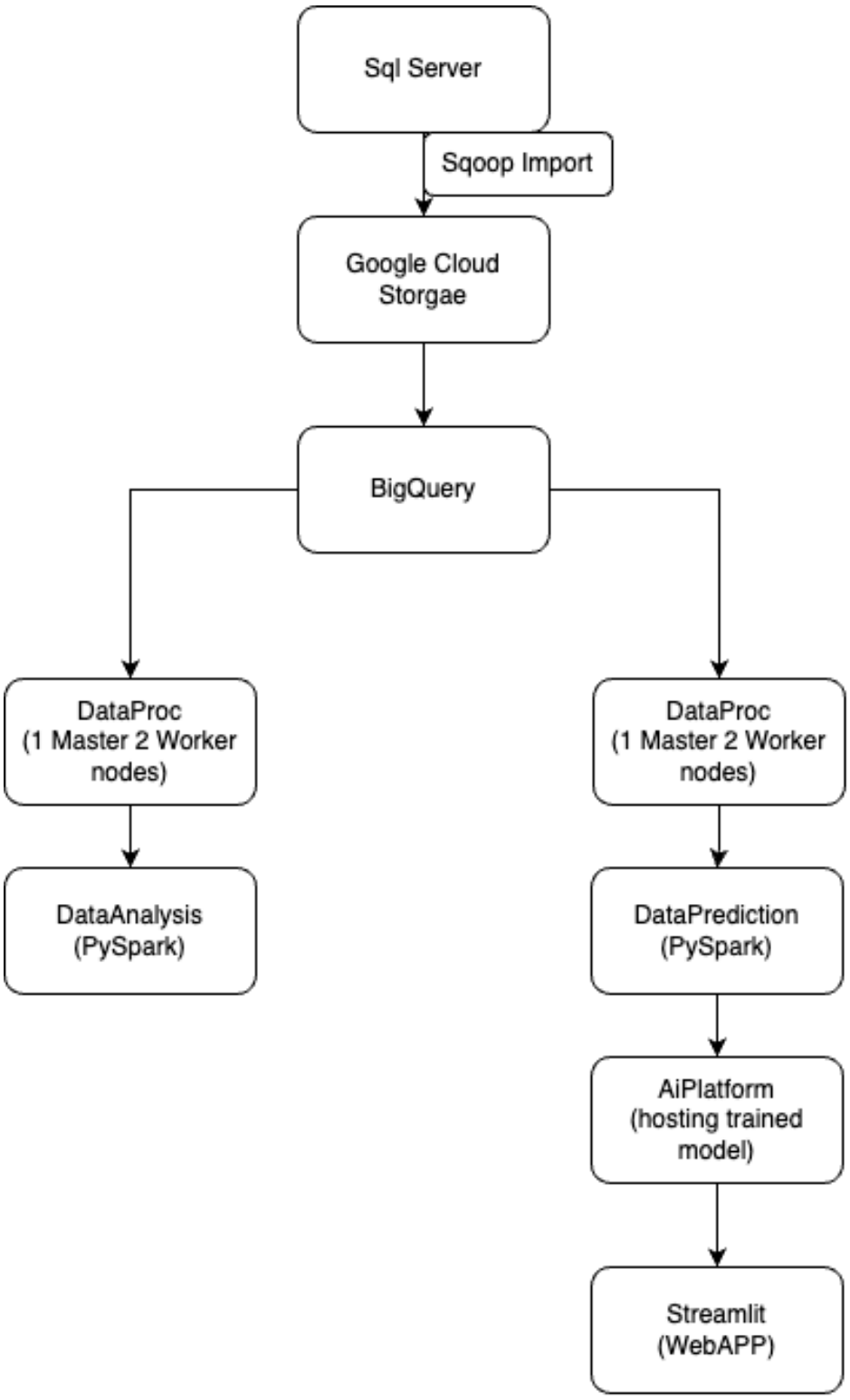
7. Deployment :

    - We were able to deploy our above trained Logistic regression model in an endpoint Using Google cloud AI Platform .
    - And accesing the hosted model via streamlit for prediction in Webapp
    - Created a webapp using Python and Streamlit and accessed the hosted model for prediction through the Webapp

Github Location for webapp code:

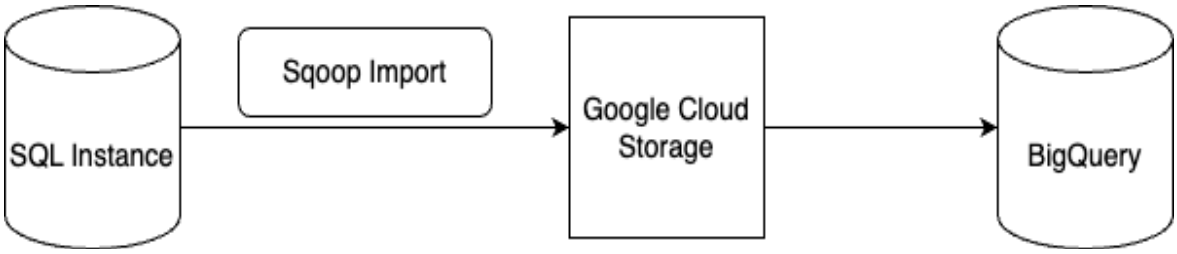
https://github.com/charanj15076/COVID_RISK_PREDICTION/tree/main/covid

Architecture and Design :



Step by Step Description of Work Flows :

Data Ingestion :



Data Ingestion Work Flow

1. First our Covid 19 covid data set is in Sql instance we imported that data from sql instance through SQOOP import to Google cloud storage as a **csv file** in to a bucket.

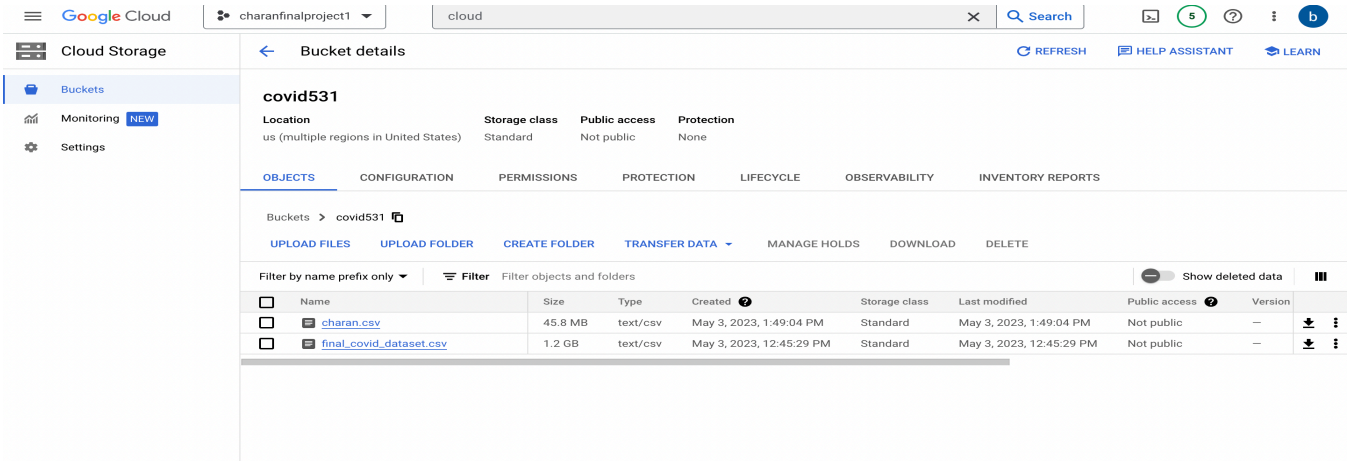
   Following are the commands used for SQOOP import :

   - We created a cluster named data-ingestion and here is the command(job) that needs to be executed in google shell:

     ```
     $ gcloud dataproc clusters create data-ingestion \
     --region us-centraleast1 \
     --master-machine-type n1-standard-2 \
     --worker-machine-type n1-standard-2 \
     --num-workers 2 \
     --image-version 1.2-debian9 \
     --properties
     dataproc:dataproc.conscrypt.provider.enable=false
     ```

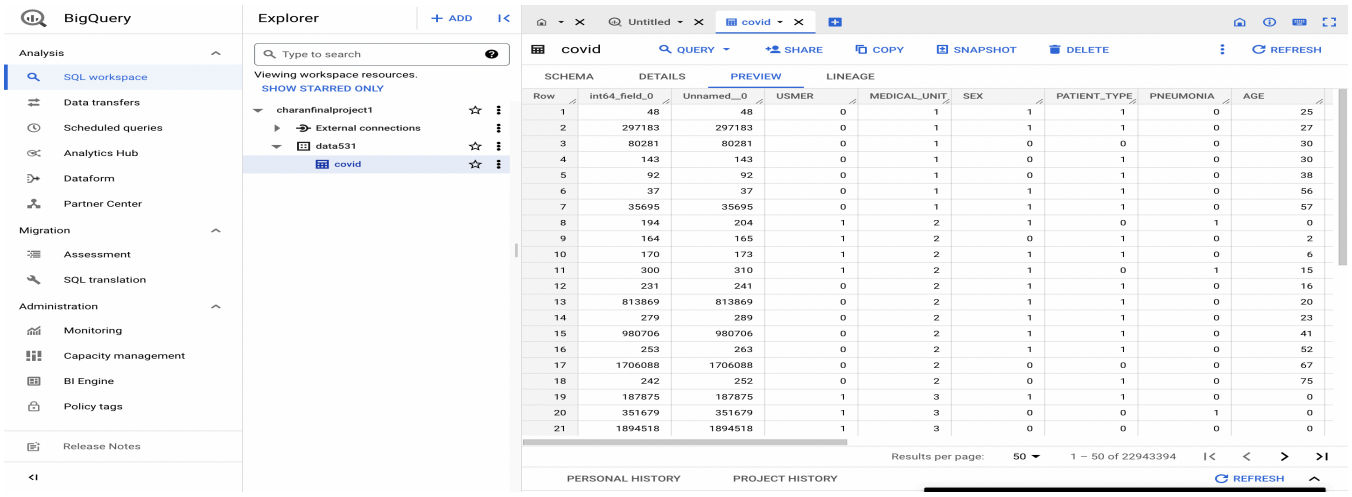
- Next Run sqoop import in that cluster and lastly delete it after import:

```
$ gcloud dataproc jobs submit hadoop \
--region us-centraleast1 \
--cluster data-ingestion \
--class org.apache.sqoop.Sqoop \
--jars $JARS \
        \
import \
-Dorg.apache.sqoop.splitter.allow_text_splitter=true \
-Dmapreduce.job.user.classpath.first=true \
--connect “gsutil:mysql://covid531/tb1 \
--username balajisaicharan9690 \
--password  \
--target-dir gs://covidbucket531/final_datset\
--delete-target-dir \
--query """SELECT * FROM transaction “””\
--as-csvdatafile \
-m 1.
```



Snapshot of CSV file after SQOOP Import

2. Next we import CSV file to BigQuery for initial analysis like changing schema and removing some unwanted coloumns.



Snapshot of dataset after loading into BigQuery

Data Analysis:

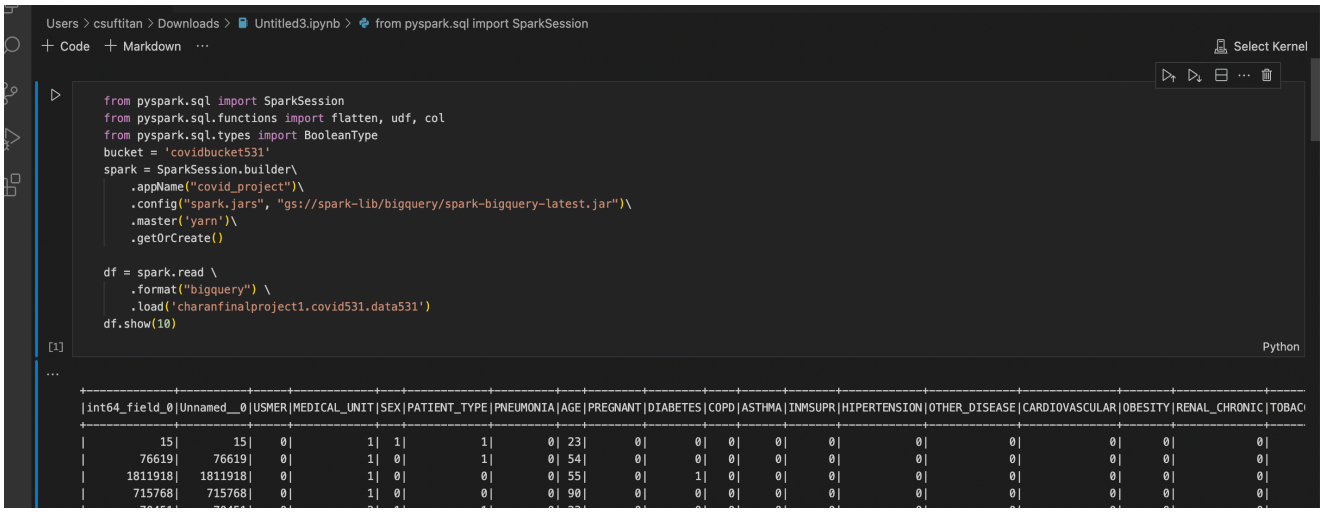
3. As the Next step we Started Cleaning our data and analyzing our data using Pyspark to do this we first created a cluster in Google cloud platform using DataProc.

   Data Cleaning code URL

   https://github.com/charanj15076/COVID_RISK_PREDICTION/blob/main/GCP_Cluster_Datacleaning.ipynb

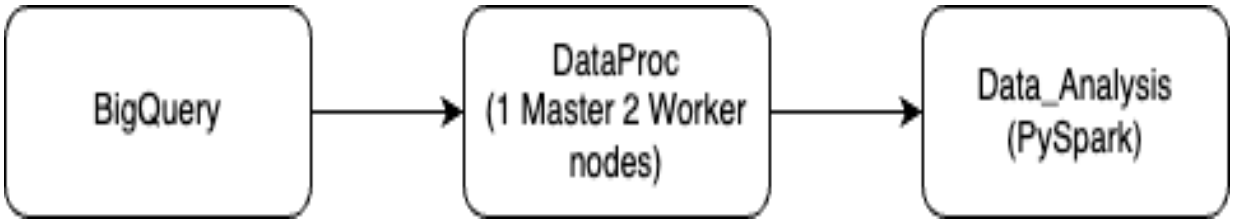
4. And Data is Loaded from BigQuery directly into the Cluster using SparkSession.

   Code for connecting BigQuery and Cluster connection

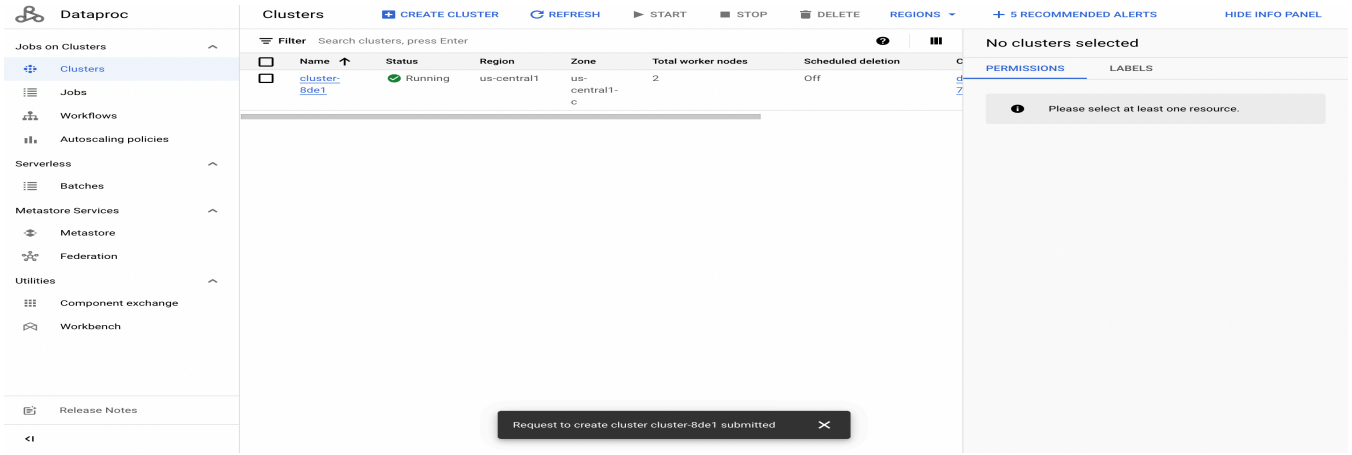


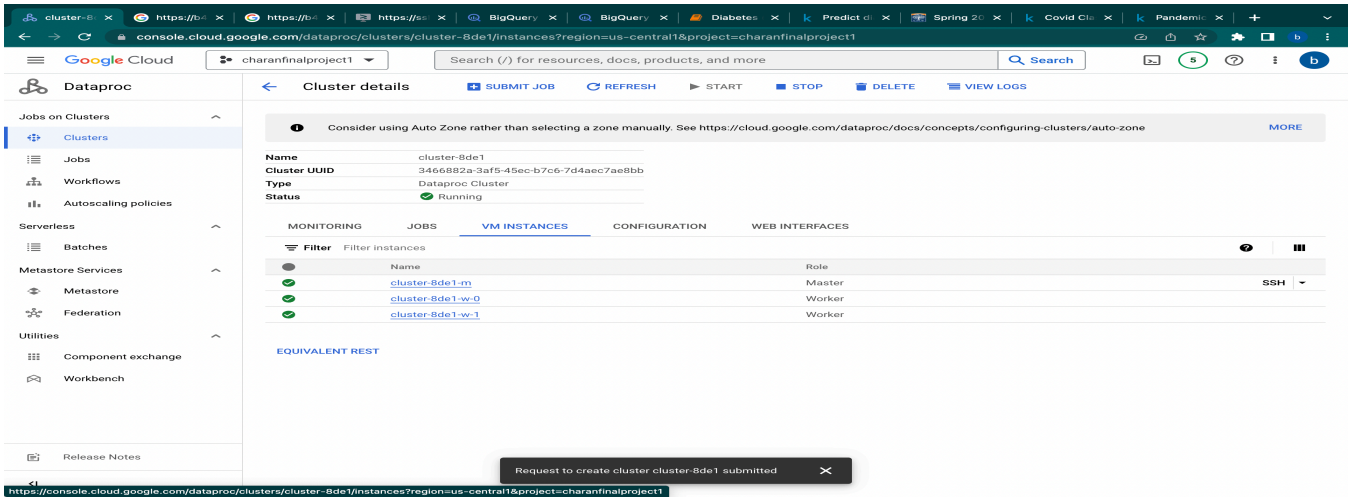
.

Data Analysis Work Flow:



- Created a cluster of 1 master and 2 worker nodes in DataProc.
  Snapshot of cluster in DataProc (Google cloud platform)



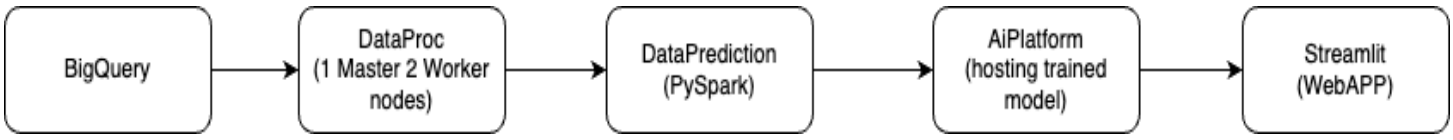


Snapshot of 1 master and 2 worker nodes

- And then using this cluster we opened a spark shell and carried our Data Cleaning and Analysis with in the cluster.

Data Analysis Code URL:
https://github.com/charanj15076/COVID_RISK_PREDICTION/blob/main/GCP_Cluster_DataAnalysis.ipynb

Data Prediction and Deployment:



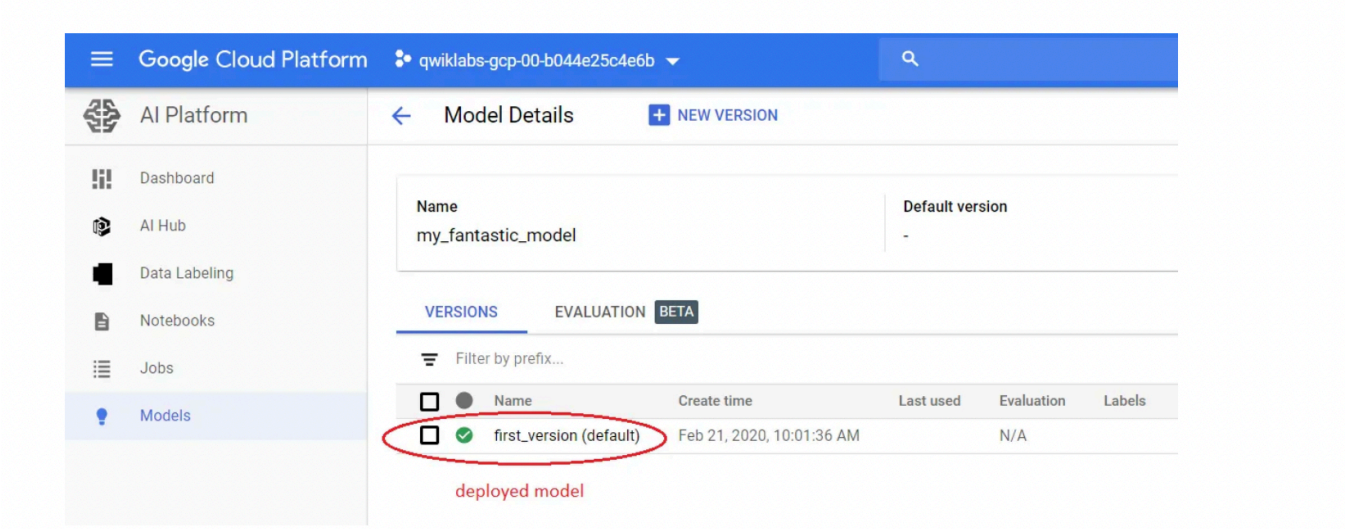
Data Prediction and Deployment Work Flow

5. Same as we created a cluster for data analysis, we created a cluster for data prediction and Submitted a **Spark JOB** for training our model and saving that model into google cloud storage.

   Data Prediction Code URL:
   https://github.com/charanj15076/COVID_RISK_PREDICTION/blob/main/Gcp_Cluster_Prediction.ipynb

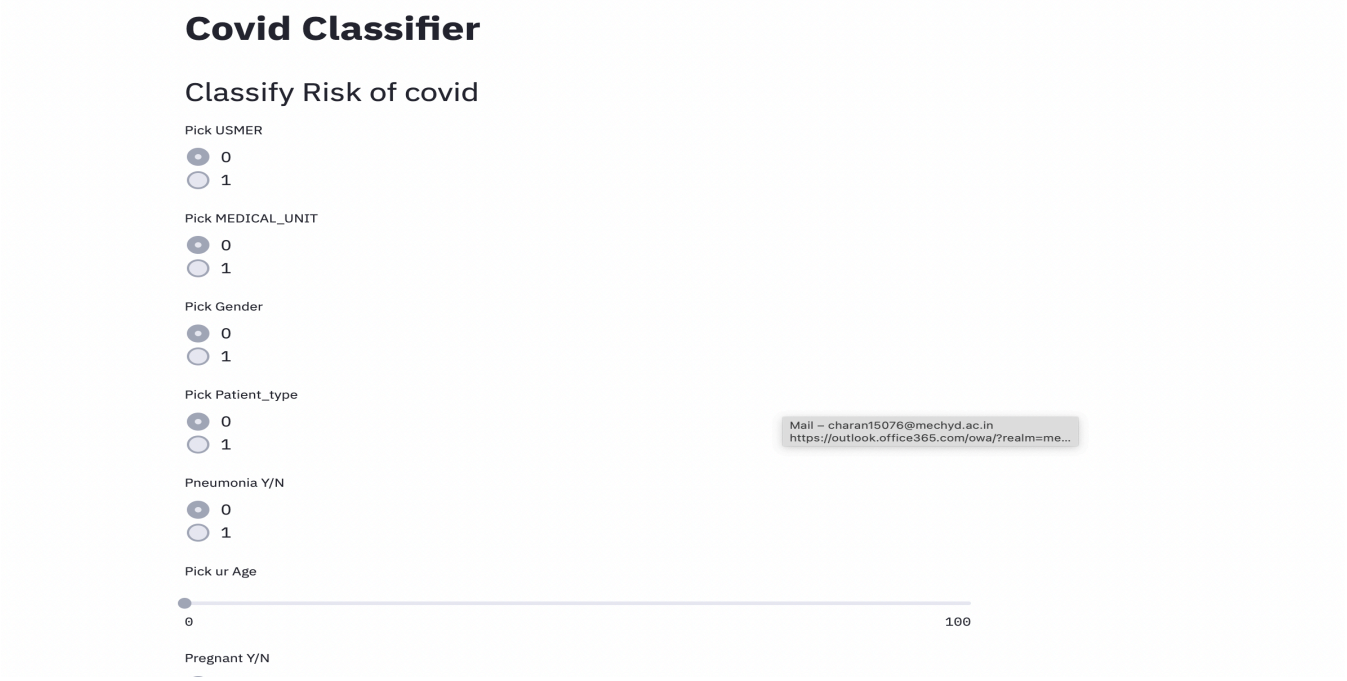
Deployment Instructions:

6. After the model is Saved in a bucket We used AI platform in GCP for hosting our model in an endpoint .



Snapshot of deployed model in an endpoint

7. Test the hosted model by making appropriate changes in the local webapp connecting the gcp and running Streamlit run app.py.



Snapshot of webapp

Github Location of code and webapp:

https://github.com/charanj15076/COVID_RISK_PREDICTION.git

code for creating webapp of our project is in Covid directory in the same repo.

Steps to run application:

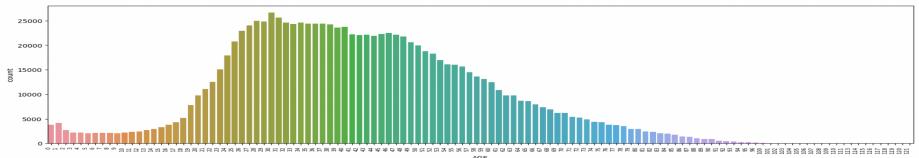
1. Login to the Google Cloud Platform (GCP) application , create a new project.
2. As our Data is in Sql instance we have to run sqoop jobs to import our data from sql to GCS bucket as csv file
3. After storing Csv file in GCS bucket search for BigQuery and create a database.
4. From that Database create a table and import that csv file from bucket to this table using gcp console.
5. Open DataProc and create a cluster with 1 master and 2 worker nodes and enable component gateway while creating cluster.
6. When the cluster is created run following codes uploaded in git ,on pyspark shell for Data Cleaning , Data Analysis and Data Prediction.
7. After running data prediction spark job host your model using AI platform in GCP
8. And download keys to access that hosted model and make appropriate changes in webapp
9. And lastly run the following command to start webapp.
10. Streamlit run app.py this invokes webapp .

Test Results:

a) Results of Data Analysis

- A plot showing us the distribution of cases over age.

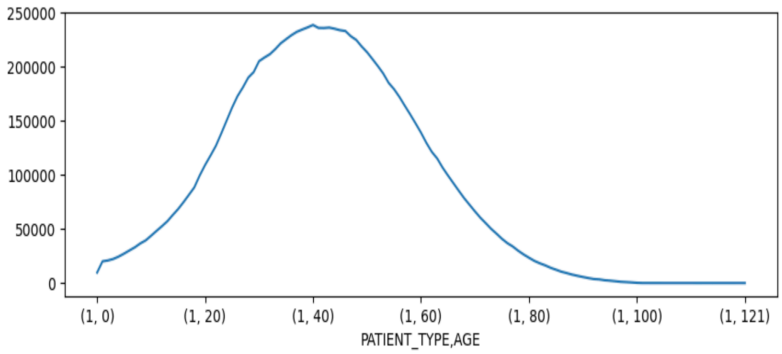
```
In [36]: data6.groupby('SEX')['DEATH'].value_counts()
Out[36]: SEX  DEATH
         0    1        6911514
              0        5941602
         1    0        5966206
              1        4124072
         Name: DEATH, dtype: int64

In [ ]: # Female deaths are  more than male deaths

In [38]: # distribution of cases over age.
         import seaborn as sns
         data6.head()
         plt.figure(figsize=(20,5))
         plt.xticks(rotation=90, horizontalalignment='right',fontweight='light',fontsize='x-small' )
         sns.countplot(x='AGE', data=data)
Out[38]: <AxesSubplot:xlabel='AGE', ylabel='count'>
```



Diabetes (2).ipynb Show All

- Which age group of Patients we sent home mostly after Medication.

```
In [44]: plt.rcParams["figure.figsize"] = (10,3)
         df = data6[data6['PATIENT_TYPE']==1.0]
         df.groupby('PATIENT_TYPE').AGE.value_counts().sort_index().plot.line()
Out[44]: <AxesSubplot:xlabel='PATIENT_TYPE,AGE'>
```



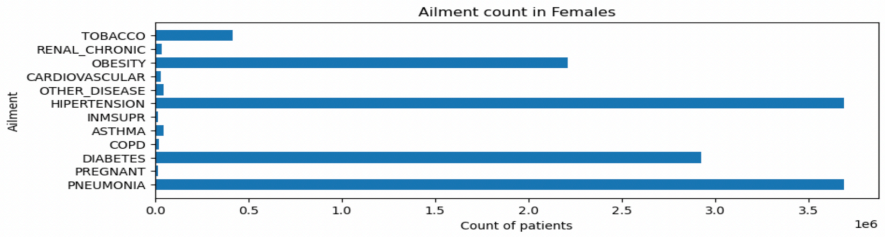

```
In [ ]: #Patients of age 20-50 were sent home mostly.

In [47]: df1 = data6[data6['CLASIFFICATION_FINAL'].isin([1.0,2,0,3.0])]
         df1.groupby('CLASIFFICATION_FINAL')['PATIENT_TYPE'].value_counts()
         # Here, 1 indicates sent home and 0 indicates hospitalised
```

Diabetes (2).ipynb Show All

- Prominent diseases among males and females that caused death.
- For Females

```
almt=[]
for i in med_hist.columns:
    ctr.append(len(df[df[i]==1]))
    almt.append(i)
plt.title('Ailment count in Females')
plt.xlabel('Count of patients')
plt.ylabel('Ailment')
plt.barh(y=almt, width=ctr)
```

Out[31]: <BarContainer object of 12 artists>



In [ ]:

In [ ]:

In [ ]:

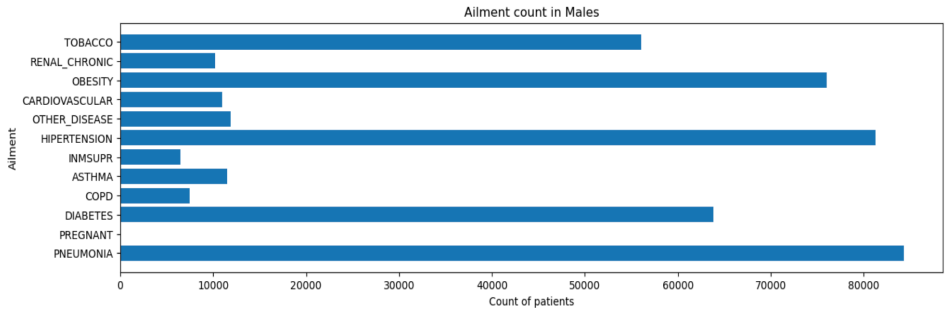
Diabetes (2).ipynb

Show All

- For males:

In [66]:
```
# Which diseases are most prominent in males.
df = data[data['SEX']== 0]
ctr=[]
almt=[]
for i in med_hist.columns:
    ctr.append(len(df[df[i]==1]))
    almt.append(i)
plt.title('Ailment count in Males')
plt.xlabel('Count of patients')
plt.ylabel('Ailment')
plt.barh(y=almt, width=ctr)
```

Out[66]: <BarContainer object of 12 artists>



In [67]:
```
df = data[data['SEX']==1]
ctr=[]
almt=[]
for i in med_hist.columns:
```

Diabetes (2).ipynb

Show All

- Percentage of people around 12 to 30 having weak immunity

In [30]:
```
#Weak immune system was a major reason for infection in young people. Let us check if our dataset reflects the same.
x=len(data6[(data6['INMSUPR']==1.0) & (data6['AGE'].isin(np.arange(15.0,30.0)))])
y=len(data6[data6['INMSUPR']==1.0])
print(x/y*100)
```

```
11.05399212737207
```

In [32]:
```
x=len(data6[(data6['INMSUPR']==1.0) & (data6['AGE'].isin(np.arange(30.0,120.0)))])
y=len(data6[data6['INMSUPR']==1.0])
print(x/y*100)
```

```
81.12331825392162
```

In [ ]:
```
#81.1% people having weak immune system are of age above 30!!
```

In [35]:
```
xyz = pd.DataFrame(data6.groupby('PATIENT_TYPE')['DEATH'].value_counts())
xyz
```

Out[35]:

		DEATH
PATIENT_TYPE	DEATH	
0	1	9617639
	0	2831490
1	0	9076318
	1	1417947

In [ ]:
```
# We can clearly see, a smaller percentage
# of patients sent home have died and a larger percentage of hospitalised have died.
```

b) Results of Data Prediction:

After Experimenting with lot of classifiers in SparkML like RandomForests, Decision Trees and Naïve Bayes. We were able to achieve higher accuracy with Logistic Regression.

Accuracy : 97.7%

```
recall 0.97853352957126
precision 0.9745297796795462
```

```
confusion matrix :

[[2158142   47344
  56405.  2323317]]
```

And thus accuracy is on Test Data , First we Split our Data into 80% for Training and 20% for Testing and on this testing Set we achieved this accuracy.

[10]:
```
from pyspark.ml.classification import LogisticRegression

lr = LogisticRegression(featuresCol = 'features' , labelCol = 'DEATH', maxIter=5)
lrModel = lr.fit(training_data)
```

```
23/05/03 23:02:21 WARN com.github.fommil.netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemB
LAS
23/05/03 23:02:21 WARN com.github.fommil.netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
```

[11]:
```
rf_predictions = lrModel.transform(test_data)
```

[12]:
```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

multi_evaluator = MulticlassClassificationEvaluator(labelCol = 'DEATH', metricName = 'accuracy')
print('Logistic Regression Accuracy:', multi_evaluator.evaluate(rf_predictions))
```

```
[Stage 17:=================================================>      (8 + 1) / 9]
Logistic Regression Accuracy: 0.9772444727883738
```