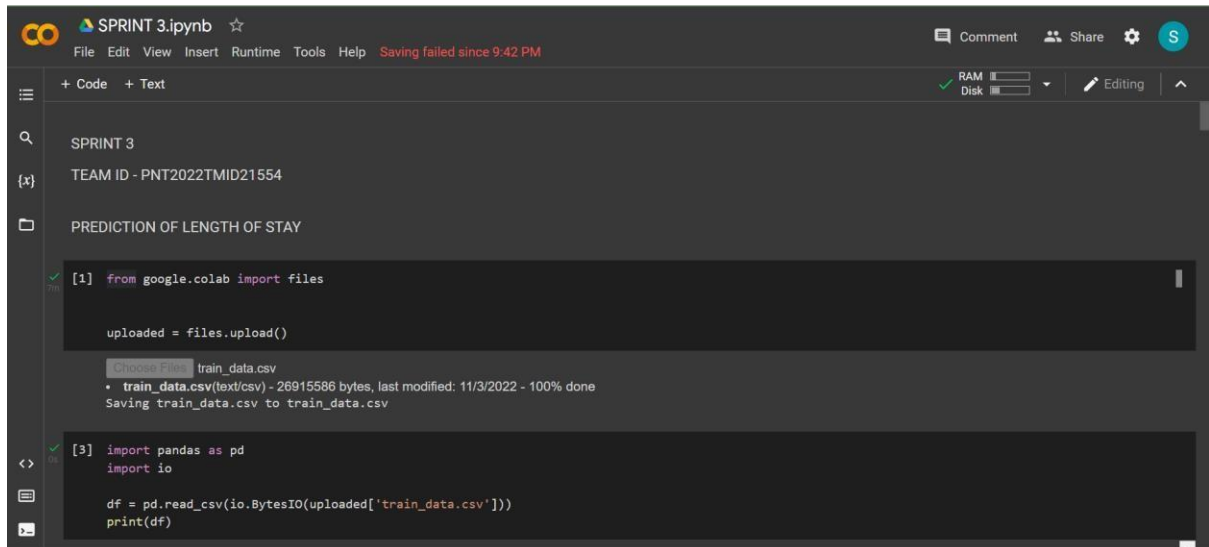


Sprint 3

TEAM ID - PNT2022TMID14193

Prediction of Length of Stay

1. Uploading Necessary files



The screenshot shows a Jupyter Notebook titled "SPRINT 3.ipynb". The left sidebar contains a search bar, a file explorer showing "SPRINT 3" and "PREDICTION OF LENGTH OF STAY", and a code editor. The main area displays two code cells. The first cell, labeled [1], contains the code to import the 'files' module from 'google.colab' and upload a file named 'train_data.csv'. A confirmation message indicates the file is 26915586 bytes and has been saved. The second cell, labeled [3], contains code to import 'pandas' as 'pd' and 'io' from 'pandas', then read the CSV file into a DataFrame 'df' and print it.

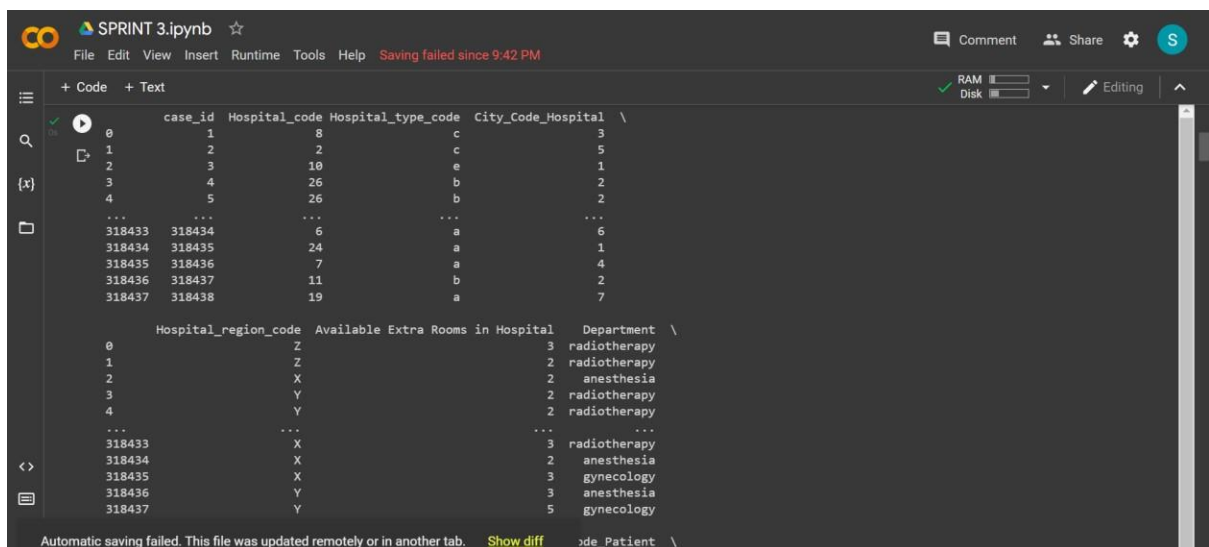
```
[1] from google.colab import files

uploaded = files.upload()

Choose File train_data.csv
• train_data.csv(text/csv) - 26915586 bytes, last modified: 11/3/2022 - 100% done
Saving train_data.csv to train_data.csv

[3] import pandas as pd
import io

df = pd.read_csv(io.BytesIO(uploaded['train_data.csv']))
print(df)
```

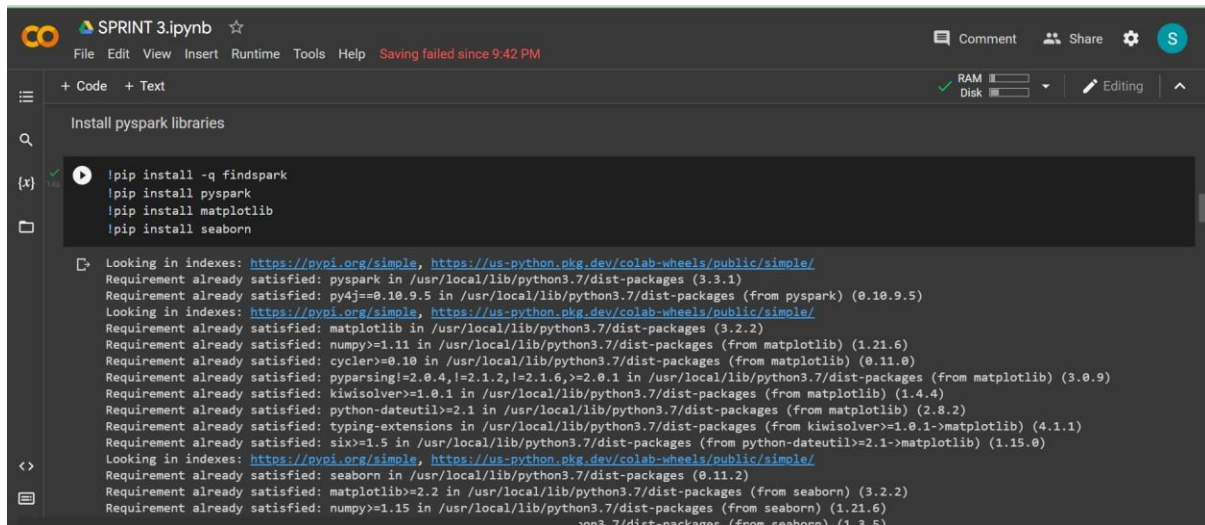


The screenshot shows the same Jupyter Notebook interface, but the second code cell now displays the output of the DataFrame 'df'. The output is a table with columns: 'case_id', 'Hospital_code', 'Hospital_type_code', 'City_Code_Hospital', 'Hospital_region_code', 'Available Extra Rooms in Hospital', and 'Department'. The data is organized into two sections, one for 'case_id' and one for 'Hospital_region_code'.

case_id	Hospital_code	Hospital_type_code	City_Code_Hospital	Hospital_region_code	Available Extra Rooms in Hospital	Department
0	1	8	c	Z	3	radiotherapy
1	2	2	c	Z	2	radiotherapy
2	3	10	e	X	2	anesthesia
3	4	26	b	Y	2	radiotherapy
4	5	26	b	Y	2	radiotherapy
...
318433	318434	6	a	X	3	radiotherapy
318434	318435	24	a	X	2	anesthesia
318435	318436	7	a	Y	3	gynecology
318436	318437	11	b	Y	3	anesthesia
318437	318438	19	a	Y	5	gynecology

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

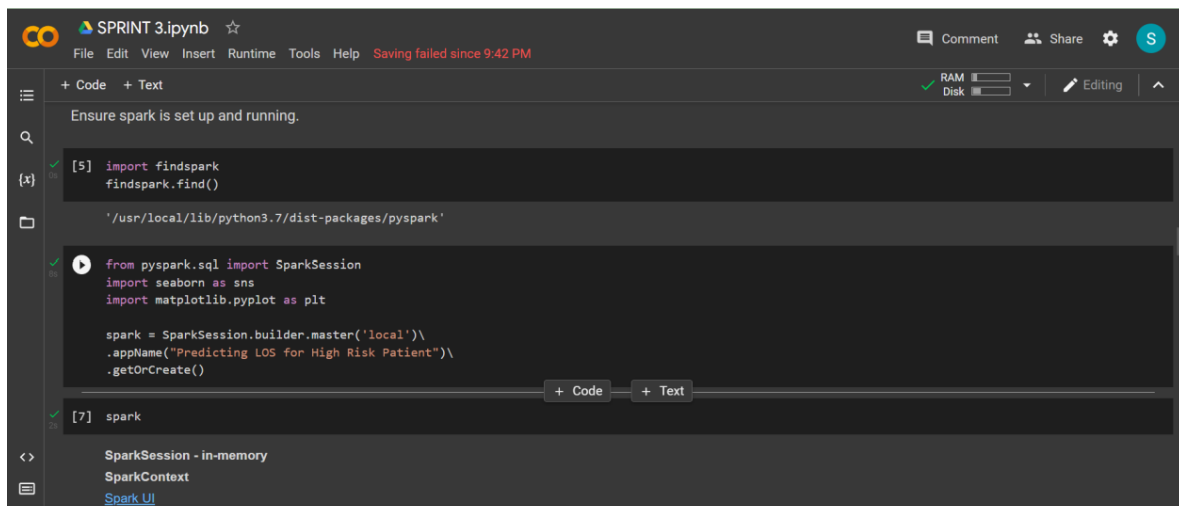
Installing pyspark Libraries



```
!pip install -q findspark
!pip install pyspark
!pip install matplotlib
!pip install seaborn
```

Looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-wheels/public/simple/>
Requirement already satisfied: pyspark in /usr/local/lib/python3.7/dist-packages (3.3.1)
Requirement already satisfied: py4j==0.10.9.5 in /usr/local/lib/python3.7/dist-packages (from pyspark) (0.10.9.5)
Looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-wheels/public/simple/>
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (3.2.2)
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.7/dist-packages (from matplotlib) (1.21.6)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib) (0.11.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib) (3.0.9)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib) (1.4.4)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from kiwisolver>=1.0.1->matplotlib) (4.1.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.1->matplotlib) (1.15.0)
Looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-wheels/public/simple/>
Requirement already satisfied: seaborn in /usr/local/lib/python3.7/dist-packages (0.11.2)
Requirement already satisfied: matplotlib>=2.2 in /usr/local/lib/python3.7/dist-packages (from seaborn) (3.2.2)
Requirement already satisfied: numpy>=1.15 in /usr/local/lib/python3.7/dist-packages (from seaborn) (1.21.6)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from seaborn) (2.8.2)

Ensuring spark is setup and it is running



```
[5] import findspark
findspark.find()

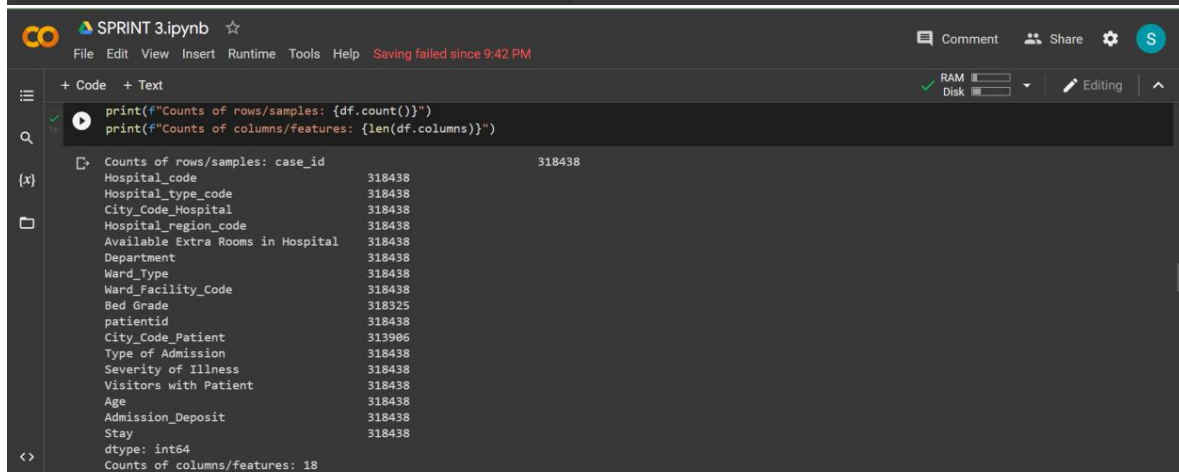
'/usr/local/lib/python3.7/dist-packages/pyspark'

from pyspark.sql import SparkSession
import seaborn as sns
import matplotlib.pyplot as plt

spark = SparkSession.builder.master('local')\
    .appName("Predicting LOS for High Risk Patient")\
    .getOrCreate()

[7] spark
```

SparkSession - in-memory
SparkContext
[Spark UI](#)



```
print(f"Counts of rows/samples: {df.count()}")
print(f"Counts of columns/features: {len(df.columns)}")
```

Counts of rows/samples: case_id 318438
Hospital_code 318438
Hospital_type_code 318438
City_Code_Hospital 318438
Hospital_region_code 318438
Available Extra Rooms in Hospital 318438
Department 318438
Ward_Type 318438
Ward_Facility_Code 318438
Bed Grade 318325
patientid 318438
City_Code_Patient 313906
Type of Admission 318438
Severity of Illness 318438
Visitors with Patient 318438
Age 318438
Admission_Deposit 318438
Stay 318438
dtype: int64
Counts of columns/features: 18

Using Machine Learning Algorithm and Principal Component Analysis(PCA)

The screenshot displays two consecutive code cells from a Jupyter Notebook titled 'SPRINT 3.ipynb'. The first cell, titled 'Machine Learning', defines a list of input variables and a target label. The second cell, titled 'Principal Component Analysis(PCA)', imports the PCA class from pyspark.ml.feature and creates a PCA object with k=10. The third cell, titled 'Standardization', imports the StandardScaler class and creates a scaler object. The fourth cell, titled 'Virtualization', creates a Pipeline object. The fifth cell, titled 'Correlation Matrix', displays the correlation matrix of the data. The sixth cell, titled 'Decision Tree', displays the decision tree model.

```
[18] input_variable = ['hospital', 'hospital_type', 'hospital_city', 'hospital_region', 'available_extra_rooms_in_hospital',  
                    'bed_grade', 'city_code_patient', 'patient_visitors', 'admission_deposit',  
                    'department_index', 'ward_facility_index', 'ward_type_index', 'illness_severity_index',  
                    'type_of_admission_index']  
  
label = ['stay_days_index']
```

Principal Component Analysis(PCA)

```
[21] from pyspark.ml.feature import PCA  
  
pca = PCA(k=10, inputCol="features", outputCol="pcaFeatures")
```

Standardization

```
[22] from pyspark.ml.feature import StandardScaler  
  
scaler = StandardScaler(inputCol="pcaFeatures", outputCol="scaledFeatures",  
                        withStd=True, withMean=False)
```

```
[27] pipeline = Pipeline(stages=[])
```

Virtualization

Correlation Matrix

Decision Tree

```
[37] df.corr().style.background_gradient(cmap='coolwarm').set_precision(2)
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

Decision Tree Algorithm for prediction

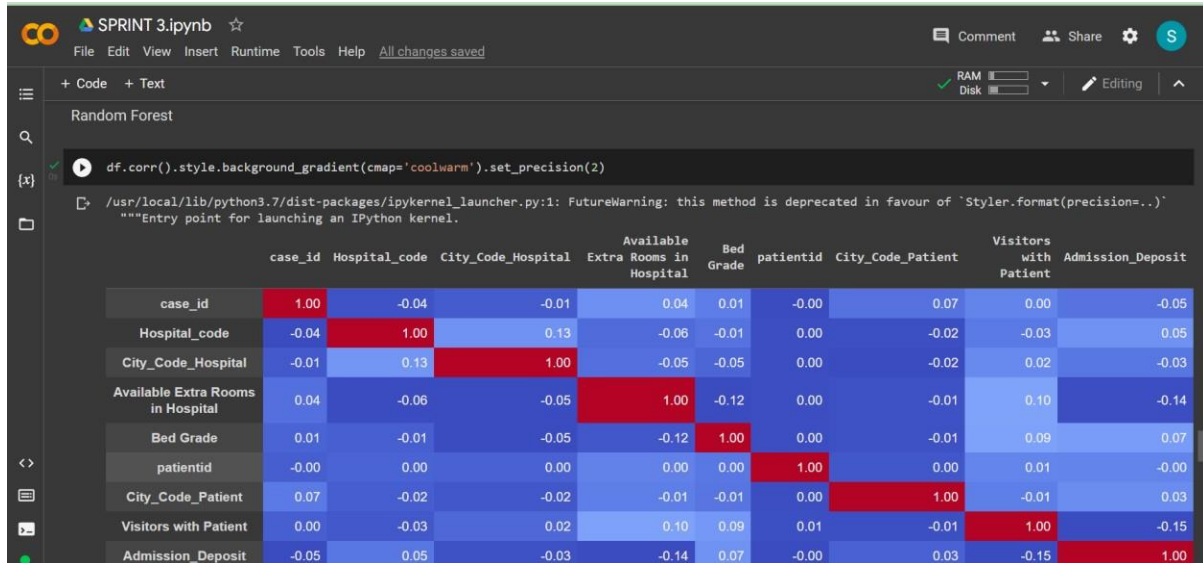
The screenshot displays a code cell from a Jupyter Notebook titled 'SPRINT 3.ipynb'. The code cell shows the correlation matrix of the data, which is a heatmap. The heatmap shows the correlation between various features, with the diagonal elements being 1.00. The features are: case_id, Hospital_code, City_Code_Hospital, Available Extra Rooms in Hospital, Bed Grade, patientid, City_Code_Patient, Visitors with Patient, and Admission_Deposit.

```
df.corr().style.background_gradient(cmap='coolwarm').set_precision(2)
```

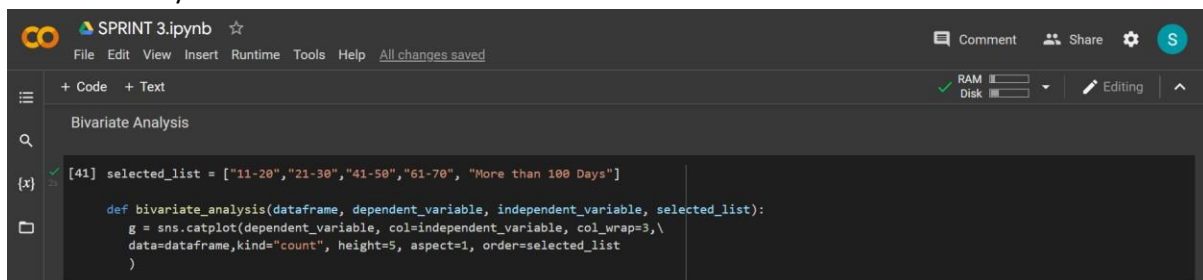
case_id Hospital_code City_Code_Hospital Available Extra Rooms in Hospital Bed Grade patientid City_Code_Patient Visitors with Patient Admission_Deposit

	case_id	Hospital_code	City_Code_Hospital	Available Extra Rooms in Hospital	Bed Grade	patientid	City_Code_Patient	Visitors with Patient	Admission_Deposit
case_id	1.00	-0.04	-0.01	0.04	0.01	-0.00	0.07	0.00	-0.05
Hospital_code	-0.04	1.00	0.13	-0.06	-0.01	0.00	-0.02	-0.03	0.05
City_Code_Hospital	-0.01	0.13	1.00	-0.05	-0.05	0.00	-0.02	0.02	-0.03
Available Extra Rooms in Hospital	0.04	-0.06	-0.05	1.00	-0.12	0.00	-0.01	0.10	-0.14
Bed Grade	0.01	-0.01	-0.05	-0.12	1.00	0.00	-0.01	0.09	0.07
patientid	-0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.01	-0.00
City_Code_Patient	0.07	-0.02	-0.02	-0.01	-0.01	0.00	1.00	-0.01	0.03
Visitors with Patient	0.00	-0.03	0.02	0.10	0.09	0.01	-0.01	1.00	-0.15
Admission_Deposit	-0.05	0.05	-0.03	-0.14	0.07	-0.00	0.03	-0.15	1.00

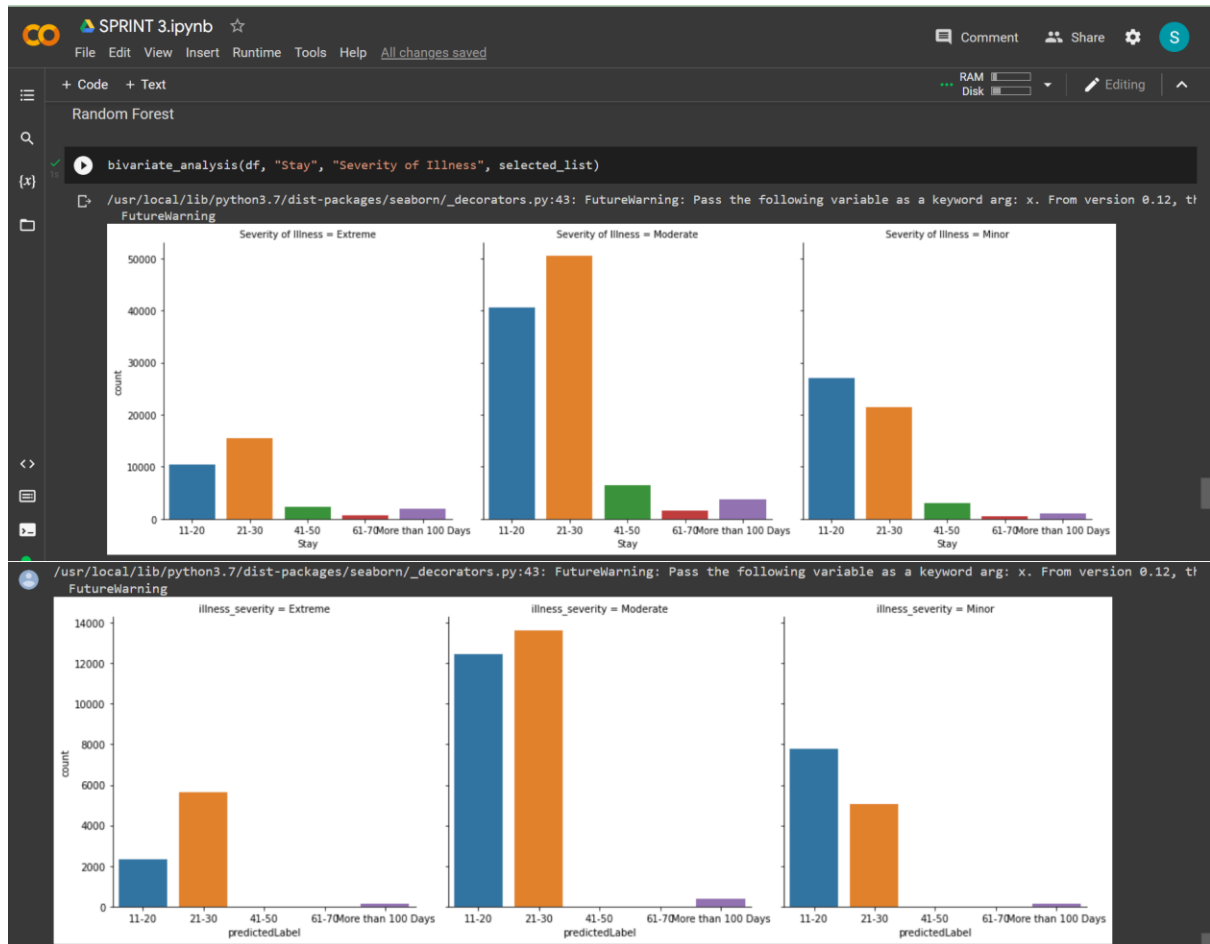
Random Forest



Bivariate Analysis



Random Forest to predict Length of Stay



Decision Tree to predict Length of Stay



