Capstone Project on Unlocking Healthcare Trends

Data Analysis on Healthcare

Presented by Mamatha Koutam

Contents:

- Abstract
- Problem Statement
- Methodology
- Result
- ***CONCLUSION***

Abstract:

This project aims to analyze healthcare data to identify key factors influencing patient outcomes, predict disease progression, and optimize treatment strategies. The results will help improve healthcare services, reduce costs, and enhance patient satisfaction.

Problem Statement:

Background: Inefficient analysis of healthcare data leads to poor patient outcomes, increased costs, and decreased satisfaction.

Objective: Unlock insights from healthcare data to improve patient care.

Scope: Focus on key healthcare metrics (e.g., accuracy score, precision score, recall score) from various machine learning models.

Methodology:

Data Collection: Retrieve data from the provided dataset.

Data Preparation: Clean and prepare data, handling missing values, and standardizing formats.

Analysis Techniques: Machine learning modeling (using techniques like DTC, RFC, XgBoost, and PCA).

Tools: Python (using libraries like pandas and sklearn) for modeling.

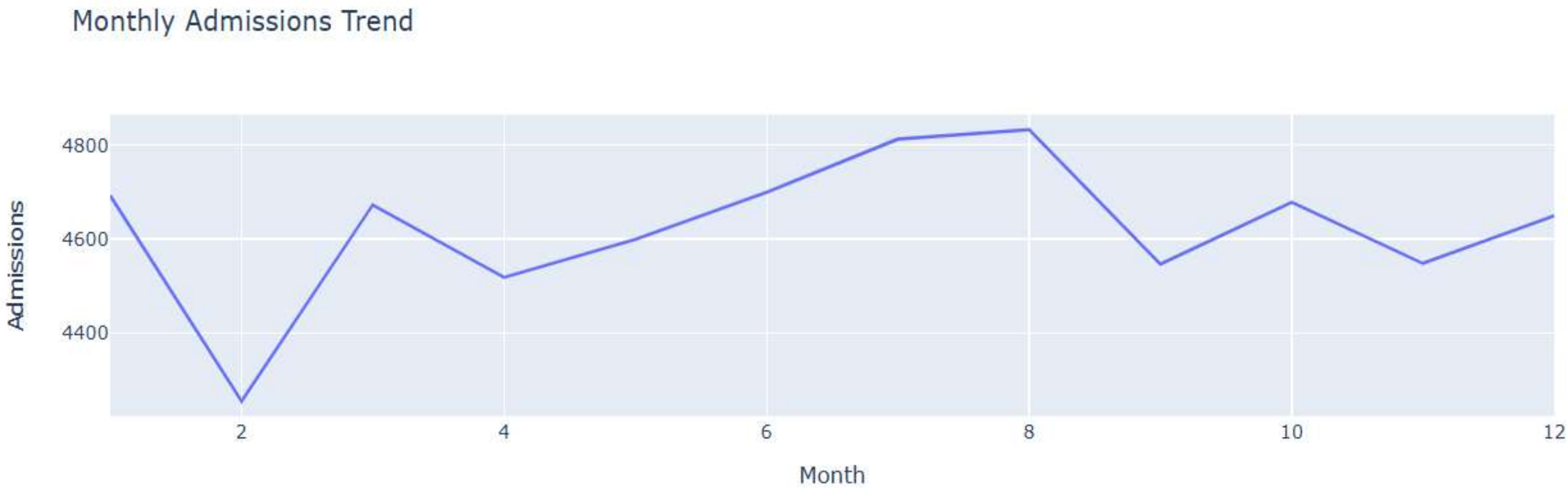
Result:

Accurate prediction of patient outcomes using machine learning models.

Identification of key factors influencing disease progression.

Optimized treatment strategies based on data insights.

Monthly Hospital Admission Trends:



Importing Data :

"Importing data is crucial as it allows us to access and utilize existing information,

saving time and resources.

- Raw data is storing into another variable I . e df

Importing data

```
data= pd.read_csv("/content/drive/MyDrive/Colab Notebooks/healthcare_dataset.csv")
data= pd.DataFrame(data)
```

```
df = data.copy(deep = True)
```

- In this ,we are checking raw data that to top 5 records by using .head() function from pandas Library

```
df.head()
```

	Name	Age	Gender	Blood Type	Medical Condition	Date of Admission	Doctor	Hospital	Insurance Provider	Billing Amount	Room Number	Admission Type	Discharge Date	Medication	Tes Result
0	Bobby JacksOn	30	Male	B-	Cancer	2024-01-31	Matthew Smith	Sons and Miller	Blue Cross	18856.281306	328	Urgent	2024-02-02	Paracetamol	Norm
1	LesLie TErRy	62	Male	A+	Obesity	2019-08-20	Samantha Davies	Kim Inc	Medicare	33643.327287	265	Emergency	2019-08-26	Ibuprofen	Inconclusi
2	DaNnY sMitH	76	Female	A-	Obesity	2022-09-22	Tiffany Mitchell	Cook PLC	Aetna	27955.096079	205	Emergency	2022-10-07	Aspirin	Norm
3	andrEw waTtS	28	Female	O+	Diabetes	2020-11-18	Kevin Wells	Hernandez Rogers and Vang,	Medicare	37909.782410	450	Elective	2020-12-18	Ibuprofen	Abnorm
4	adrIENNE bEll	43	Female	AB+	Cancer	2022-09-19	Kathleen Hanna	White-White	Aetna	14238.317814	458	Urgent	2022-10-09	Penicillin	Abnorm

- In this, we are checking shape of data means that data consist of 15 attributes and 55,500 rows.
  By help of .shape function from pandas Library

```
#Check the shape of data
print(f'The Training Dataset has {df.shape[0]} rows and {df.shape[1]} columns.')
```

```
The Training Dataset has 55500 rows and 15 columns.
```

- Here , we can observe data types of each attribute and is there any missing values or not.
  So in this data , we don't have any missing values and we have to modify datatype of DOA , DD
  These are can easily fetch by help of .info() function i.e. from pandas Library.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 55500 entries, 0 to 55499
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Name                55500 non-null  object 
 1   Age                 55500 non-null  int64  
 2   Gender              55500 non-null  object 
 3   Blood Type          55500 non-null  object 
 4   Medical Condition   55500 non-null  object 
 5   Date of Admission   55500 non-null  object 
 6   Doctor              55500 non-null  object 
 7   Hospital            55500 non-null  object 
 8   Insurance Provider  55500 non-null  object 
 9   Billing Amount      55500 non-null  float64
 10  Room Number         55500 non-null  int64  
 11  Admission Type      55500 non-null  object 
 12  Discharge Date      55500 non-null  object 
 13  Medication          55500 non-null  object 
 14  Test Results        55500 non-null  object 
dtypes: float64(1), int64(2), object(12)
memory usage: 6.4+ MB
```

- .describe() will helps to observe as shown below and can explain in below for numerical and categorical.

```
df.describe()
```

	Age	Date of Admission	Billing Amount	Room Number	Discharge Date
count	55500.000000	55500	55500.000000	55500.000000	55500
mean	51.539459	2021-11-01 01:02:22.443243008	25539.316097	301.134829	2021-11-16 13:15:20.821621504
min	13.000000	2019-05-08 00:00:00	-2008.492140	101.000000	2019-05-09 00:00:00
25%	35.000000	2020-07-28 00:00:00	13241.224652	202.000000	2020-08-12 00:00:00
50%	52.000000	2021-11-01 00:00:00	25538.069376	302.000000	2021-11-17 00:00:00
75%	68.000000	2023-02-03 00:00:00	37820.508436	401.000000	2023-02-18 00:00:00
max	89.000000	2024-05-07 00:00:00	52764.276736	500.000000	2024-06-06 00:00:00
std	19.602454	NaN	14211.454431	115.243069	NaN

```
df.describe(include= "object").T
```

	count	unique	top	freq
Name	55500	40235	michael williams	24
Gender	55500	2	Male	27774
Blood Type	55500	8	A-	6969
Medical Condition	55500	6	Arthritis	9308
Doctor	55500	40341	Michael Smith	27
Hospital	55500	39876	LLC Smith	44
Insurance Provider	55500	5	Cigna	11249
Admission Type	55500	3	Elective	18655
Medication	55500	5	Lipitor	11140
Test Results	55500	3	Abnormal	18627

Observations:

1.Patient Age Range:

1. Patients' ages range from 13 to 89 years, with an average age of approximately 52 years.

2.Hospital Room Capacity:

1. The hospital offers a range of rooms, from 101 to 500, ensuring flexibility in patient accommodation.

3.Temporal Coverage:

1. Data spans from May 8, 2019, to May 7, 2024, providing a comprehensive five-year view of patient admissions.

4.Admission Types:

1. Patients enter the hospital through three main admission routes:
    1. Emergency
    2. Elective
    3. Transfer

5.Blood Type Distribution:

1. Patients exhibit various blood types, with A- being the most prevalent.

6.Hospital Distribution:

1. The dataset encompasses admissions from 44 hospitals, with LLC Smith being the most frequent.

7.Doctor Distribution:

1. Among the 27 doctors recorded in the dataset, Michael Smith attends to the highest number of patients.

- In this, we are separating independent and dependent variables for model building

Separating independent variables and dependent variable¶

```
X = df.iloc[:, df.columns != 'Test Results']
y = df[['Test Results']]
```

```
X.shape , y.shape
```

```
((55500, 14), (55500, 1))
```

- For Classification , in the target, we have to check how many classes and those classes are balanced or not if not we should make balance by help of imbalance learn library.

```
from imblearn.over_sampling import RandomOverSampler
```

```
X_re , y_re = RandomOverSampler(random_state=42).fit_resample(X,y)
```

```
X_re.shape , y_re.shape
```

```
((55881, 14), (55881, 1))
```

```
y_re.value_counts()
```

Test Results	count
Abnormal	18627
Inconclusive	18627
Normal	18627

dtype: int64

- Before Train the model we have convert data into numeric then computer can able to fetch the data.

```
from sklearn.preprocessing import LabelEncoder

for col in X_re.columns:
    if X_re[col].dtypes == 'object':
        X_re[col] = LabelEncoder().fit_transform(X_re[col])
```

- Feature elimination: if the data contains un necessary feature that means its not impact to prediction then we can remove feature .
  * Hence , it is called “Feature elimination”.

```
X_re.drop('Name' ,axis = 1, inplace=True)
```

```
X_re.head()
```

	Age	Gender	Blood Type	Medical Condition	Date of Admission	Doctor	Hospital	Insurance Provider	Billing Amount	Room Number	Admission Type	Discharge Date	M
0	30	1	5	2	1706659200000000000	26612	29933	1	18856.281306	328	2	1706832000000000000	
1	62	1	0	5	1566259200000000000	33648	16012	3	33643.327287	265	1	1566777600000000000	
2	76	0	1	5	1663804800000000000	37828	5473	0	27955.096079	205	1	1665100800000000000	
3	28	0	6	3	1605657600000000000	22511	12317	3	37909.782410	450	0	1608249600000000000	
4	43	0	2	2	1663545600000000000	21259	33598	0	14238.317814	458	2	1665273600000000000	

- After Completed EDA , we should split the data into train and test by help of sklearn library

```
] from sklearn.model_selection import train_test_split
```

```
X_train , X_test , y_train ,y_test = train_test_split(X_re,y_re,test_size=0.2,random_state=42)
print(f"X_train{X_train.shape}")
print(f"X_test{X_test.shape}")
print(f"y_train{y_train.shape}")
print(f"y_test{y_test.shape}")
```

```
X_train(44704, 13)
X_test(11177, 13)
y_train(44704, 1)
y_test(11177, 1)
```

- This UDfunction trains a given machine learning model on training data, evaluates its performance on testing data, and prints various metrics such as accuracy, Cohen's Kappa, classification report, and confusion matrix. It returns the trained model, accuracy, Cohen's Kappa, and time taken to train the model, providing a comprehensive evaluation of the model's performance.

```python
import time
from sklearn.metrics import accuracy_score, cohen_kappa_score, confusion_matrix, classification_report
```

```python
def run_model(model, X_train, y_train, X_test, y_test, verbose=True):
    t0=time.time()
    if verbose == False:
        model.fit(X_train,y_train, verbose=0)
    else:
        model.fit(X_train,y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    coh_kap = cohen_kappa_score(y_test, y_pred)
    cm = confusion_matrix(y_test , y_pred)
    time_taken = time.time()-t0
    print("Accuracy = {}".format(accuracy))
    print("Cohen's Kappa = {}".format(coh_kap))
    print("Time taken = {} minutes".format(time_taken/60))
    print("classification_report is\n ",classification_report(y_test,y_pred,digits=5))
    print(f"confusion matrix \n {cm}")



    return model, accuracy, coh_kap, time_taken
```

- The model's performance is relatively low, with an accuracy of 0.3427 and Cohen's Kappa of 0.0151, indicating a high degree of randomness in the predictions. The classification report and confusion matrix show significant misclassifications across all classes, with no clear strengths or weaknesses in the model's predictions.
  * In that way, we will check for multiple models and compare which model is giving good performance that can say best model for this data.

```
Accuracy = 0.3426679788852107
Cohen's Kappa = 0.015078539640364874
Time taken = 6.156691579023997 minutes
classification_report is
                precision    recall  f1-score   support

    Abnormal    0.34305   0.24807   0.28793      3749
Inconclusive    0.33915   0.45482   0.38856      3674
      Normal    0.34727   0.32738   0.33704      3754

    accuracy                        0.34267     11177
   macro avg    0.34316   0.34342   0.33784     11177
weighted avg    0.34319   0.34267   0.33750     11177

confusion matrix
 [[ 930 1644 1175]
 [ 868 1671 1135]
 [ 913 1612 1229]]
```

❖ Here , we can see on different models with and their performances.

Report

	Model	Accuracy Score	Precision Score	Recall Score	Kappa Score	f1-score
0	LogisticReg with Penalty L1	0.342913	0.343398	0.306	0.015600	0.3080
1	Pruned DecisionTreeClassifier	0.348913	0.353398	0.306	0.025600	0.3380
2	RandomForestClassifier	0.404891	0.403398	0.416	0.105600	0.4080
3	GradientBoostingClassifier	0.444891	0.443398	0.456	0.165600	0.4480
4	XgBoostClassifier	0.445891	0.453398	0.466	0.167600	0.4580
5	AdaBoostClassifier	0.335891	0.355398	0.406	0.004976	0.3780
6	XgBoostClassifier with StandardScalar	0.445891	0.439800	0.446	0.167600	0.4500
7	XgBoostClassifier with MinMaxdScalar	0.445891	0.439800	0.446	0.167600	0.4500
8	RFC with MinMaxScalar By GridSearchCV	0.448891	0.459800	0.446	0.176000	0.4500
9	RFC with StandardScalar By GridSearchCV	0.428891	0.459800	0.446	0.176000	0.4500
10	XgBoostClassifier By GridSearchCV	0.428891	0.439800	0.446	0.126000	0.4440
11	RFC with SelectKBest FeatureSelection	0.372300	0.388000	0.426	0.059600	0.3844
12	RFC with Forward Selection	0.448891	0.459800	0.446	0.176000	0.4500

- Here, is a continuation of the comparison.
  * By comparing all model we can finalize best model .

9	RFC with StandardScalar By GridSearchCV	0.428891	0.459800	0.446	0.176000	0.4500
10	XgBoostClassifier By GridSearchCV	0.428891	0.439800	0.446	0.126000	0.4440
11	RFC with SelectKBest FeatureSelection	0.372300	0.388000	0.426	0.059600	0.3844
12	RFC with Forward Selection	0.448891	0.459800	0.446	0.176000	0.4500
13	RFC with Backward Selecton	0.447800	0.459800	0.446	0.176000	0.4500
14	RFC with SelectFromModel FS	0.447130	0.459800	0.446	0.176000	0.4500
15	RFC with RFE FS	0.441300	0.459800	0.446	0.176000	0.4500
16	RFC with PCA	0.448130	0.459800	0.446	0.176000	0.4500

CONCLUSION:

- Best Performing Model: RFC with MinMaxScalar By GridSearchCV (Row 8) and RFC with Forward Selection (Row 12) achieve the highest accuracy score of 0.448891, followed closely by other models.
- I have tried with Ensemble methods (RFC, XgBoost) outperform single models (Logistic Regression, Decision Tree). Feature selection/ engineering techniques (MinMaxScalar, StandardScalar, Forward Selection, Backward Selection, SelectFromModel FS) improve model performance.
- Hyperparameter tuning using GridSearchCV enhances model accuracy. Dimensionality reduction using PCA also performs well.
- The data is not extremely bad, as some models achieve relatively good performance.

Thank You

For your attention