This work deals with a model to predict the healthcare cost using the fusion of ML and DL. The method we have proposed is Deep Hyrbid Learning. Tabular data with a regression problem to solve always inclines towards machine learning. Deep Learning is much popular for the imagess but not restricted only to it. DL is always complex for tabular data but what if we can use both ml and dl. My idea is not to ensemble which makes it much more complex rather use the benefits of both the ideologies. ML is simple, faster to model and learn but deep learning isnt. On the other hand Neural networks has the ability to short down the important features in a very huge dataset on its own without any feature selection work unlike we do in Machine Learning. Even though there are techniques around to select features for ML models. Deep learning is much effective in coming with the important features. So we use deep learning model on the data to extract important features and use those features on a machine learning model to get a better accuracy

Random Forest - Phase 1

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

# To ignore any warnings
import warnings
warnings.filterwarnings("ignore")
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Saved successfully!

```
Drive/RF+NN/Inpatient_sparcs.csv',header=None)
```

```
df.shape
```

```
(2343570, 34)
```

```
df.head().T
```

	0	1	2	3	4
0	Hospital Service Area	Hudson Valley	Hudson Valley	Hudson Valley	Hudson Valley
1	Hospital County	Westchester	Westchester	Westchester	Westchester
2	Operating Certificate Number	5903001	5903001	5903001	5903001
3	Permanent Facility Id	001061	001061	001061	001061
4	Facility Name	Montefiore Mount Vernon Hospital	Montefiore Mount Vernon Hospital	Montefiore Mount Vernon Hospital	Montefiore Mount Vernon Hospital
5	Age Group	30 to 49	50 to 69	30 to 49	50 to 69
6	Zip Code - 3 digits	NaN	105	105	105
7	Gender	M	M	F	F
8	Race	White	White	White	White
9	Ethnicity	Not Span/Hispanic	Spanish/Hispanic	Unknown	Not Span/Hispanic
10	Length of Stay	21	8	6	4
11	Type of Admission	Elective	Emergency	Emergency	Emergency
12	Patient Disposition	Home or Self Care	Skilled Nursing Home	Court/Law Enforcement	Skilled Nursing Home
13	Discharge Year	2017	2017	2017	2017

```
df.head()
```

Saved successfully!

			2	3	4	5	6	7	8	9	...
0	Hospital Service Area	Hospital County	Operating Certificate Number	Permanent Facility Id	Facility Name	Age Group	Zip Code - 3 digits	Gender	Race	Ethnicity	...
1	Hudson Valley	Westchester	5903001	001061	Montefiore Mount Vernon Hospital	30 to 49	NaN	M	White	Not Span/Hispanic	...
2	Hudson Valley	Westchester	5903001	001061	Montefiore Mount Vernon Hospital	50 to 69	105	M	White	Spanish/Hispanic	...
3	Hudson Valley	Westchester	5903001	001061	Montefiore Mount Vernon Hospital	30 to 49	105	F	White	Unknown	...
4	Hudson Valley	Westchester	5903001	001061	Montefiore Mount Vernon Hospital	50 to 69	105	F	White	Not Span/Hispanic	...

5 rows × 34 columns

APR Risk of

```
headers = ["Hospital_Service_Area","Hospital_County","Operating_Certificate_Number","Permanent_Facility_Id","Facility_Name","Age_Group","Zip_

df.columns = headers

pd.options.display.max_columns = 40

df = df.iloc[1:, :]

df.head(10)
```

	Hospital_Service_Area	Hospital_County	Operating_Certificate_Number	Permanent_Facility_Id	Facil
1	Hudson Valley	Westchester	5903001	001061	M Mou
2	Hudson Valley	Westchester	5903001	001061	M Mou
3	Hudson Valley	Westchester	5903001	001061	M Mou
4	Hudson Valley	Westchester	5903001	001061	M Mou
5	Hudson Valley	Westchester	5903001	001061	M Mou
6	Hudson Valley	Westchester	5903001	001061	M Mou
7	Hudson Valley	Westchester	5903001	001061	M Mou
8	Hudson Valley	Westchester	5903001	001061	M Mou
9	Hudson Valley	Westchester	5903001	001061	M Mou
		Westchester	5903001	001061	M Mou

Saved successfully!

```
df['Length_of_Stay'] = pd.to_numeric(df['Length_of_Stay'],errors = 'coerce')
df['Total_Costs'] = pd.to_numeric(df['Total_Costs'],errors = 'coerce')
df['Total_Charges'] = pd.to_numeric(df['Total_Charges'],errors = 'coerce')
```

```
df.dtypes
```

```
Hospital_Service_Area                  object
Hospital_County                        object
Operating_Certificate_Number           object
Permanent_Facility_Id                  object
Facility_Name                          object
Age_Group                              object
Zip_Code                               object
Gender                                 object
Race                                   object
Ethnicity                              object
Length_of_Stay                        float64
Type_of_Admission                      object
Patient_Disposition                    object
Discharge_Year                         object
CCS_Diagnosis_Code                     object
CCS_Diagnosis_Description              object
CCS_Procedure_Code                     object
CCS_Procedure_Description              object
APR-DRG_Code                           object
APR_DRG_Description                    object
APR_MDC_Code                           object
APR_MDC_Description                    object
APR_Severity_of_Illness_Code           object
APR_Severity_of_Illness_Description    object
APR_Risk_of_Mortality                  object
APR_Medical_Surgical_Description       object
Payment_Typology1                      object
Payment_Typology2                      object
Payment_Typology3                      object
Birth_Weight                           object
Abortion_Edit_Indicator                object
Emergency_Department_Indicator         object
Total_Charges                         float64
Total_Costs                           float64
dtype: object
```

```
df.isnull().sum(axis = 0)
```

```
Hospital_Service_Area                     5155
Hospital_County                           5155
Operating_Certificate_Number              5155
Permanent_Facility_Id                     5155
Facility_Name                                0
Age_Group                                    0
Zip_Code                                 39019
Gender                                       0
Race                                         0
Ethnicity                                    0
Length_of_Stay                            1739
Type_of_Admission                            0
Patient_Disposition                          0
Discharge_Year                               0
CCS_Diagnosis_Code                           0
CCS_Diagnosis_Description                    0
CCS_Procedure_Code                           0
CCS_Procedure_Description                    0
APR-DRG_Code                                 0
APR_DRG_Description                          0
APR_MDC_Code                                 0
APR_MDC_Description                          0
APR_Severity_of_Illness_Code                 0
APR_Severity_of_Illness_Description        240
APR_Risk_of_Mortality                      240
APR_Medical_Surgical_Description             0
Payment_Typology1                            0
Payment_Typology2                       878722
Payment_Typology3                      1737244
Birth_Weight                           2115685
Abortion_Edit_Indicator                      0
Emergency_Department_Indicator               0
Total_Charges                                0
Total_Costs                                  0
dtype: int64
```

Saved successfully!

number of missing data

Double-click (or enter) to edit

```
del df['Birth_Weight']
```

```
del df['Payment_Typology3']
```

```
del df['Payment_Typology2']
```

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

```
Hospital_Service_Area                   5155
Hospital_County                         5155
Operating_Certificate_Number            5155
Permanent_Facility_Id                   5155
Facility_Name                              0
Age_Group                                  0
Zip_Code                               39019
Gender                                     0
Race                                       0
Ethnicity                                  0
Length_of_Stay                          1739
Type_of_Admission                          0
Patient_Disposition                        0
Discharge_Year                             0
CCS_Diagnosis_Code                         0
CCS_Diagnosis_Description                  0
CCS_Procedure_Code                         0
CCS_Procedure_Description                  0
APR-DRG_Code                               0
APR_DRG_Description                        0
APR_MDC_Code                               0
APR_MDC_Description                        0
APR_Severity_of_Illness_Code               0
APR_Severity_of_Illness_Description      240
APR_Risk_of_Mortality                    240
APR_Medical_Surgical_Description           0
```

```
Payment_Typology1                        0
Abortion_Edit_Indicator                  0
Emergency_Department_Indicator           0
Total_Charges                            0
Total_Costs                              0
dtype: int64
```

```
df.shape
```

```
(2343569, 31)
```

Removing Rows with Null values in any of the features

```
df.dropna(subset = ["Hospital_Service_Area"], inplace=True)
```

```
df.dropna(subset = ["Hospital_County"], inplace=True)
```

```
df.dropna(subset = ["Operating_Certificate_Number"], inplace=True)
```

```
df.dropna(subset = ["Permanent_Facility_Id"], inplace=True)
```

```
df.dropna(subset = ["APR_Severity_of_Illness_Description"], inplace=True)
```

```
df.dropna(subset = ["APR_Risk_of_Mortality"], inplace=True)
```

```
df.dropna(subset = ["Zip_Code"], inplace=True)
```

```
df.dropna(subset = ["Length_of_Stay"], inplace=True)
```



```
Hospital_Service_Area                0
Hospital_County                      0
Operating_Certificate_Number         0
Permanent_Facility_Id                0
Facility_Name                        0
Age_Group                            0
Zip_Code                             0
Gender                               0
Race                                 0
Ethnicity                            0
Length_of_Stay                       0
Type_of_Admission                    0
Patient_Disposition                  0
Discharge_Year                       0
CCS_Diagnosis_Code                   0
CCS_Diagnosis_Description            0
CCS_Procedure_Code                   0
CCS_Procedure_Description            0
APR-DRG_Code                         0
APR_DRG_Description                  0
APR_MDC_Code                         0
APR_MDC_Description                  0
APR_Severity_of_Illness_Code         0
APR_Severity_of_Illness_Description  0
APR_Risk_of_Mortality                0
APR_Medical_Surgical_Description     0
Payment_Typology1                    0
Abortion_Edit_Indicator              0
Emergency_Department_Indicator       0
Total_Charges                        0
Total_Costs                          0
dtype: int64
```

No Feature has empty values now

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2302682 entries, 2 to 2343569
Data columns (total 31 columns):
```

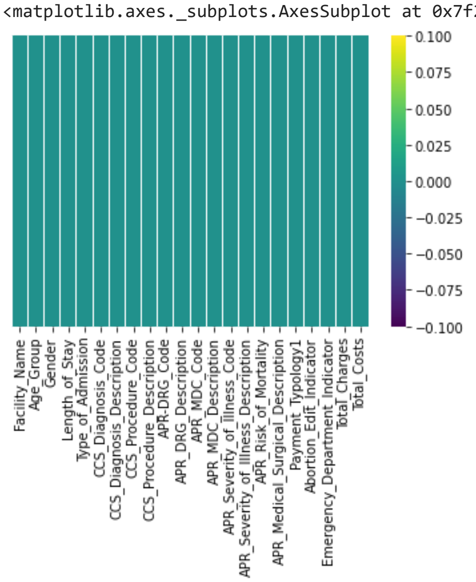
```
 #   Column                           Dtype  
---  ------                           -----  
 0   Hospital_Service_Area            object 
 1   Hospital_County                  object 
 2   Operating_Certificate_Number     object 
 3   Permanent_Facility_Id            object 
 4   Facility_Name                    object 
 5   Age_Group                        object 
 6   Zip_Code                         object 
 7   Gender                           object 
 8   Race                             object 
 9   Ethnicity                        object 
 10  Length_of_Stay                   float64
 11  Type_of_Admission                object 
 12  Patient_Disposition              object 
 13  Discharge_Year                   object 
 14  CCS_Diagnosis_Code               object 
 15  CCS_Diagnosis_Description        object 
 16  CCS_Procedure_Code               object 
 17  CCS_Procedure_Description        object 
 18  APR-DRG_Code                     object 
 19  APR_DRG_Description              object 
 20  APR_MDC_Code                     object 
 21  APR_MDC_Description              object 
 22  APR_Severity_of_Illness_Code     object 
 23  APR_Severity_of_Illness_Description  object 
 24  APR_Risk_of_Mortality            object 
 25  APR_Medical_Surgical_Description  object 
 26  Payment_Typology1                object 
 27  Abortion_Edit_Indicator          object 
 28  Emergency_Department_Indicator   object 
 29  Total_Charges                    float64
 30  Total_Costs                      float64
dtypes: float64(3), object(28)
memory usage: 562.2+ MB
```

Removal of the features which might not affect the healthcare cost

Exploring the Data

```
# Checking the Missing Values by Visualiztion
sns.heatmap(df.isnull(), yticklabels=False, cmap= "viridis")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f289f17ead0>
```

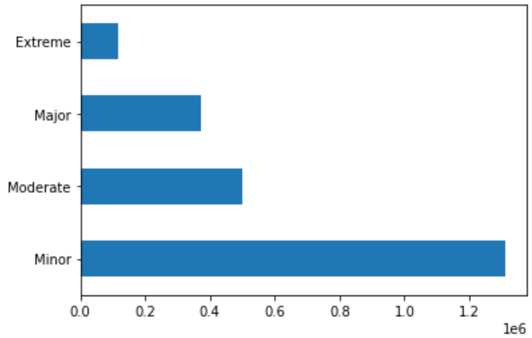



```
df['APR_Risk_of_Mortality'].value_counts()
```

```
Minor       1313337
Moderate     500735
Major        372440
Extreme      116170
Name: APR_Risk_of_Mortality, dtype: int64
```

```
df.APR_Risk_of_Mortality.value_counts().plot(kind='barh')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f289ee0f150>
```

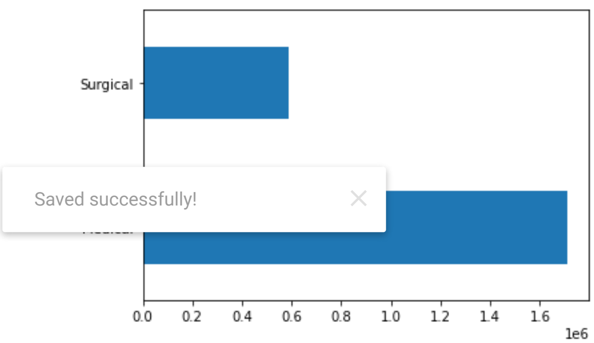



```
df['APR_Medical_Surgical_Description'].value_counts()
```

```
Medical     1713122
Surgical     589560
Name: APR_Medical_Surgical_Description, dtype: int64
```

```
df.APR_Medical_Surgical_Description.value_counts().plot(kind='barh')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f287de7aa50>
```

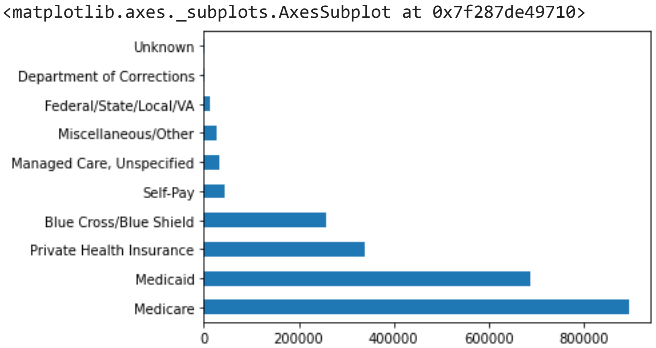



```
df['Payment_Typology1'].value_counts()
```

```
Medicare                     897426
Medicaid                     687928
Private Health Insurance     337882
Blue Cross/Blue Shield       258108
Self-Pay                      43242
Managed Care, Unspecified     31714
Miscellaneous/Other           27639
Federal/State/Local/VA        13986
Department of Corrections      2512
Unknown                        2245
Name: Payment_Typology1, dtype: int64
```

```
df.Payment_Typology1.value_counts().plot(kind='barh')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f287de49710>
```



```
df['Abortion_Edit_Indicator'].value_counts()
```

```
N    2302682
Name: Abortion_Edit_Indicator, dtype: int64
```

Removing this feature as only few diagnosis deals with pregnancy or abortion

```
del df['Abortion_Edit_Indicator']
```

```
del df['Emergency_Department_Indicator']
```

Removing the duplicated rows

```
df.duplicated().sum()
```

```
61349
```

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

	Facility_Name	Age_Group	Gender	Length_of_Stay	Type_of_Admission	CCS_Diagnosis_Code	CCS_D
2	Montefiore Mount Vernon Hospital	50 to 69	M	8.0	Emergency	099	com
3	Montefiore Mount Vernon Hospital	30 to 49	F	6.0	Emergency	161	Oth
4	Montefiore Mount Vernon Hospital	50 to 69	F	4.0	Emergency	238	
[illegible]	[illegible] Hospital	[illegible] 29	F	4.0	Emergency	002	Se
6	Montefiore Mount Vernon Hospital	50 to 69	M	3.0	Emergency	660	
...	...	...	...	...	...	...	
2343564	Hospital for Special Surgery	50 to 69	M	8.0	Elective	238	
2343565	Good Samaritan Hospital of Suffern	50 to 69	F	2.0	Elective	133	
2343566	Good Samaritan Hospital of Suffern	50 to 69	F	2.0	Emergency	101	Cor
2343568	Montefiore Med Center - Jack D Weiler Hosp of ...	70 or Older	F	2.0	Elective	25	
2343569	St James Mercy Hospital	18 to 29	F	1.0	Emergency	660	

2241333 rows × 20 columns

Saved successfully!

```
df.reset_index(inplace = True)
```

```
df
```

	index	Facility_Name	Age_Group	Gender	Length_of_Stay	Type_of_Admission	CCS_Diagnosis_Cc
0	2	Montefiore Mount Vernon Hospital	50 to 69	M	8.0	Emergency	(
1	3	Montefiore Mount Vernon Hospital	30 to 49	F	6.0	Emergency	1
2	4	Montefiore Mount Vernon Hospital	50 to 69	F	4.0	Emergency	2
3	5	Montefiore Mount Vernon Hospital	18 to 29	F	4.0	Emergency	(
4	6	Montefiore Mount Vernon Hospital	50 to 69	M	3.0	Emergency	6
...	...	...	...	...	...	...	
2302677	2343564	Hospital for Special Surgery	50 to 69	M	8.0	Elective	2
2302678	2343565	Good Samaritan Hospital of Suffern	50 to 69	F	2.0	Elective	1
2302679	2343566	Good Samaritan Hospital of Suffern	50 to 69	F	2.0	Emergency	1
[illegible]	[illegible]	Montefiore Med [illegible] D [illegible] of ...	70 or Older	F	2.0	Elective	
2302681	2343569	St James Mercy Hospital	18 to 29	F	1.0	Emergency	6

Saved successfully!

2302682 rows × 21 columns

```
del df['index']
```

```
df.head()
```

	Facility_Name	Age_Group	Gender	Length_of_Stay	Type_of_Admission	CCS_Diagnosis_Code	CCS_Diagnos
0	Montefiore Mount Vernon Hospital	50 to 69	M	8.0	Emergency	099	complicatio
1	Montefiore Mount Vernon Hospital	30 to 49	F	6.0	Emergency	161	Other disea
2	Montefiore Mount Vernon Hospital	50 to 69	F	4.0	Emergency	238	Compli proce
3	Montefiore Mount Vernon Hospital	18 to 29	F	4.0	Emergency	002	Septicemi
4	Montefiore Mount Vernon Hospital	50 to 69	M	3.0	Emergency	660	Alcohol

Removing the diagnosis description codes as they do not satisfy the integer conditions. We will encode the description soon on our own

```
columns2=['Facility_Name', 'Age_Group', 'Gender', 'Length_of_Stay', 'Type_of_Admission', 'CCS_Diagnosis_Description', 'CCS_Procedure_Descript
```

```
df2=df
```

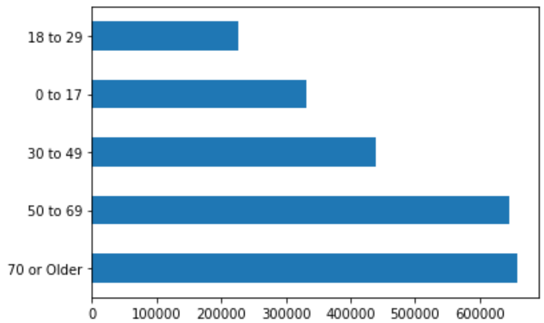
```
df=df[columns2]
```

```
df['Age_Group'].value_counts()
```

```
70 or Older    659132
50 to 69       646204
30 to 49       439330
0 to 17        331488
18 to 29       226528
Name: Age_Group, dtype: int64
```

```
df.Age_Group.value_counts().plot(kind='barh')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f287ddd5410>
```



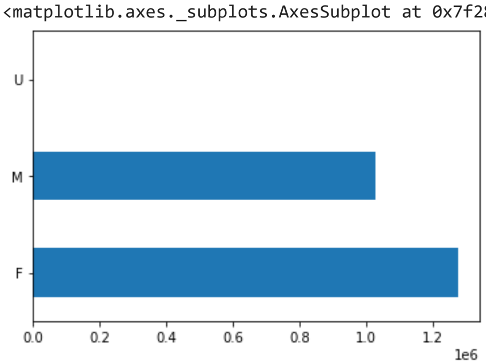

```
df['Gender'].value_counts()
```

```
F    1275423
M    1027239
```

Saved successfully!

```
df.Gender.value_counts().plot(kind='barh')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f287dcfc190>
```




```
#Different Number of Hospitals
n=len(df['Facility_Name'].unique())
print(n)
```

```
209
```

```
df.shape
```

```
(2302682, 14)
```

```
df.dtypes
```

```
Facility_Name                   object
Age_Group                       object
Gender                          object
Length_of_Stay                 float64
Type_of_Admission               object
CCS_Diagnosis_Description       object
CCS_Procedure_Description       object
```

```
APR_DRG_Description                      object
APR_MDC_Description                      object
APR_Severity_of_Illness_Description      object
APR_Risk_of_Mortality                    object
APR_Medical_Surgical_Description         object
Payment_Typology1                        object
Total_Costs                             float64
dtype: object
```

Label Encoding

```
from sklearn.preprocessing import LabelEncoder
labelenc=LabelEncoder()
```

```
df['Facility_Name_Code'] = labelenc.fit_transform(df['Facility_Name'])
```

```
df['Age_Group_Code'] = labelenc.fit_transform(df['Age_Group'])
```

```
df['Gender_Code'] = labelenc.fit_transform(df['Gender'])
```

```
df['Type_of_Admission_Code'] = labelenc.fit_transform(df['Type_of_Admission'])
```

```
df['Risk_of_Mortality_Code'] = labelenc.fit_transform(df['APR_Risk_of_Mortality'])
```

```
df['APR_DRG_Desc_Code'] = labelenc.fit_transform(df['APR_DRG_Description'])
```

```
df['APR_MDC_Desc_Code'] = labelenc.fit_transform(df['APR_MDC_Description'])
```

```
nsform(df['APR_Severity_of_Illness_Description'])
```



```
df['Surgical_Desc_Code'] = labelenc.fit_transform(df['APR_Medical_Surgical_Description'])
```

```
df['Payment_Typology1_Code'] = labelenc.fit_transform(df['Payment_Typology1'])
```

```
df['CCS_Diagnosis_Encode'] = labelenc.fit_transform(df['CCS_Diagnosis_Description'])
```

Filtering down to the important columns

```
ft_df=['Facility_Name_Code','Age_Group_Code','Gender_Code','Length_of_Stay','Type_of_Admission_Code','CCS_Diagnosis_Encode','APR_DRG_Desc_Cod
```

```
df3=df
```

```
df=df[ft_df]
```

```
df.dtypes
```

```
Facility_Name_Code          int64
Age_Group_Code              int64
Gender_Code                 int64
Length_of_Stay            float64
Type_of_Admission_Code      int64
CCS_Diagnosis_Encode        int64
APR_DRG_Desc_Code           int64
Illness_Code                int64
Risk_of_Mortality_Code      int64
Surgical_Desc_Code          int64
Payment_Typology1_Code      int64
Total_Costs               float64
dtype: object
```

Correlation

```
df.corr()
```

	Facility_Name_Code	Age_Group_Code	Gender_Code	Length_of_Stay	Type_of_Admi
Facility_Name_Code	1.000000	0.013969	-0.005848	-0.004070	
Age_Group_Code	0.013969	1.000000	0.022728	0.110151	
Gender_Code	-0.005848	0.022728	1.000000	0.055122	
Length_of_Stay	-0.004070	0.110151	0.055122	1.000000	
Type_of_Admission_Code	0.021481	-0.177003	-0.009090	0.019077	
CCS_Diagnosis_Encode	-0.004358	-0.071467	-0.083229	0.037275	
APR_DRG_Desc_Code	0.007218	-0.092314	-0.084430	0.002041	
Illness_Code	-0.021293	-0.136189	-0.043467	-0.249370	
Risk_of_Mortality_Code	-0.015636	-0.100481	-0.016017	-0.198687	
Surgical_Desc_Code	0.006665	0.121048	-0.032848	0.044950	
Payment_Typology1_Code	-0.014395	0.150420	0.012700	0.029553	
Total_Costs	0.019269	0.093593	0.037387	0.488168	

```
df.corr().style.background_gradient(cmap="Blues")
```

	Facility_Name_Code	Age_Group_Code	Gender_Code	Length_of_Stay	Type_of_Admi
Facility_Name_Code	1.000000	0.013969	-0.005848	-0.004070	
Age_Group_Code	0.013969	1.000000	0.022728	0.110151	
Gender_Code	-0.005848	0.022728	1.000000	0.055122	
[illegible]	-0.004070	0.110151	0.055122	1.000000	
Type_of_Admission_Code	0.021481	-0.177003	-0.009090	0.019077	
CCS_Diagnosis_Encode	-0.004358	-0.071467	-0.083229	0.037275	
APR_DRG_Desc_Code	0.007218	-0.092314	-0.084430	0.002041	
Illness_Code	-0.021293	-0.136189	-0.043467	-0.249370	
Risk_of_Mortality_Code	-0.015636	-0.100481	-0.016017	-0.198687	
Surgical_Desc_Code	0.006665	0.121048	-0.032848	0.044950	
Payment_Typology1_Code	-0.014395	0.150420	0.012700	0.029553	
Total_Costs	0.019269	0.093593	0.037387	0.488168	

Saved successfully!

Random Forest Model Implementation

```
Y = df['Total_Costs']
X = df.drop('Total_Costs',axis=1)
```

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.inspection import permutation_importance
from matplotlib import pyplot as plt

plt.rcParams.update({'figure.figsize': (12.0, 8.0)})
plt.rcParams.update({'font.size': 14})
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.33,random_state=12)
```

With 25 estimators

Double-click (or enter) to edit

```
rf = RandomForestRegressor(n_estimators=25)
rf.fit(X_train, Y_train)
```

```
RandomForestRegressor(n_estimators=25)
```

```
Y_pred=rf.predict(X_test)
```

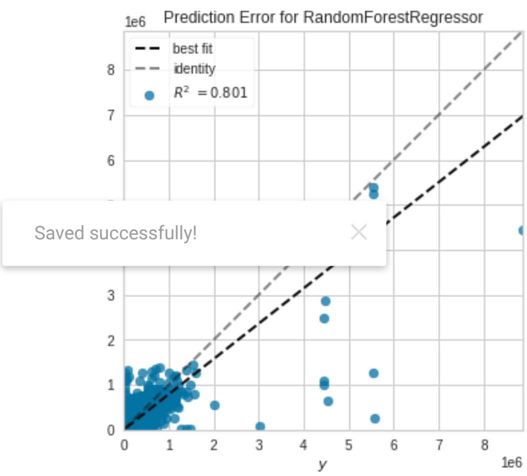
```
from sklearn.metrics import r2_score
r2_score(Y_test, Y_pred)
```

```
0.8006124452378273
```

```
from yellowbrick.datasets import load_concrete
from yellowbrick.regressor import PredictionError
```

```
model = rf
visualizer = PredictionError(model)

visualizer.fit(X_train, Y_train)  # Fit the training data to the visualizer
visualizer.score(X_test, Y_test)  # Evaluate the model on the test data
visualizer.show()
```




```
<matplotlib.axes._subplots.AxesSubplot at 0x7f287bff3d10>
```

Phase 2 - Deep Learning

- Now we have to import the preprocessed data to run neural networks on the data , the data is normalized between 0 to 1

```
df = pd.read_csv(r'/content/drive/MyDrive/RF+NN/dataset.csv')
```

```
del df['Unnamed: 0']
```

```
import tensorflow as tf
```

```
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
df=df.dropna()
X = df
y = df.pop('Total Costs')
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.50, random_state=40)
```

```
from numpy.ma.core import shape
from keras import layers
from keras.layers import Dropout
```

```
from keras.models import Sequential
from keras.layers import Dense, LSTM, TimeDistributed
from keras.wrappers.scikit_learn import KerasRegressor
from keras import optimizers

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler


def regression_model():
    # Define model
    model = Sequential()
    model.add(Dense(32*9, input_dim=X_train.shape[1], activation='relu'))
    model.add(Dense(32*9, activation='relu'))
    model.add(Dense(64*8, activation='relu'))
    model.add(Dense(1, activation='linear'))
    # Compile model
    adam = tf.keras.optimizers.Adam(lr=0.001)
    model.compile(loss='mean_squared_error', optimizer=adam,metrics=['accuracy'])

    return model

# Use KerasRegressor wrapper (from Keras to sklearn)
# The packages we use are meant to be run with sklearn models
estimator = KerasRegressor(build_fn=regression_model, validation_split = 0.2, batch_size=1000, epochs=50, verbose=0)
history = estimator.fit(X_train, y_train)
```

```
estimator.model.save('neuralmodel_2.h5')
```

```
new_model = tf.keras.models.load_model('neuralmodel_2.h5')
```

Saved successfully!

```
pre=new_model.predict(X_test)
r2=r2_score(y_test,pre)
```

```
print(r2)
```

```
0.8217285664120255
```

```
#import necessary libraries
from sklearn.metrics import mean_squared_error
from math import sqrt

#calculate RMSE
sqrt(mean_squared_error(y_test, pre))
```

```
0.343572795268529
```

```
print(history.history.keys())
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'dev'], loc='upper left')
plt.show()
```

Feature Extraction using neural network

We use various feature information gathering methods to filter down the important ones using Shap Values, Permutation Importance and Partial Dependence Plot

```
!pip install shap
```

```
Collecting shap
  Downloading shap-0.40.0-cp37-cp37m-manylinux2010_x86_64.whl (564 kB)
     |████████████████████████████████| 564 kB 36.3 MB/s
Requirement already satisfied: tqdm>4.25.0 in /usr/local/lib/python3.7/dist-packages (from shap) (4.64.0)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from shap) (1.3.5)
Requirement already satisfied: numba in /usr/local/lib/python3.7/dist-packages (from shap) (0.51.2)
Requirement already satisfied: packaging>20.9 in /usr/local/lib/python3.7/dist-packages (from shap) (21.3)
Collecting slicer==0.0.7
  Downloading slicer-0.0.7-py3-none-any.whl (14 kB)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.7/dist-packages (from shap) (1.3.0)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from shap) (1.0.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from shap) (1.4.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from shap) (1.21.6)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from packaging>20.9->shap) (3.0.8)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (from numba->shap) (57.4.0)
Requirement already satisfied: llvmlite<0.35,>=0.34.0.dev0 in /usr/local/lib/python3.7/dist-packages (from numba->shap) (0.34.0)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas->shap) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas->shap) (2022.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->pandas->shap) (1.15.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn->shap) (3.1.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn->shap) (1.1.0)
Installing collected packages: slicer, shap
Successfully installed shap-0.40.0 slicer-0.0.7
```

```
import shap
```

```
def f_wrapper(X):
    return estimator.predict(X).flatten()
```

- By SHAP

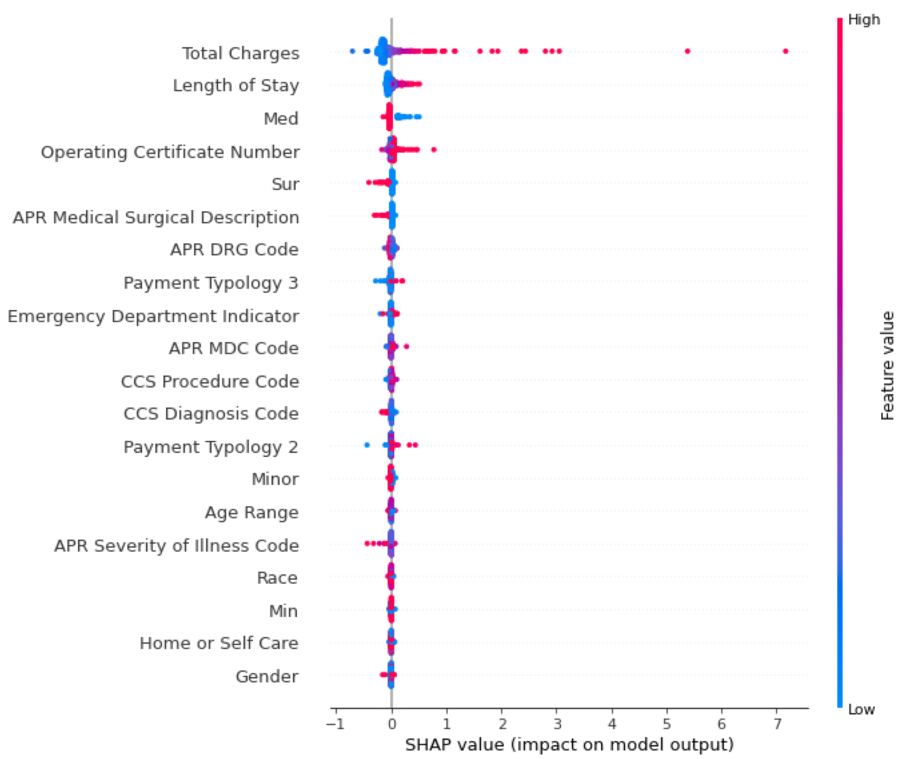
```
X_train_summary = shap.kmeans(X_train, 20)

# Compute Shap values
rapper,X_train_summary)

s
# The training set is too big so let's sample it. We get enough point to draw conclusions
X_train_sample = X_train.sample(400)
shap_values  = explainer.shap_values(X_train_sample)
shap.summary_plot(shap_values, X_train_sample)
```

Saved successfully!

100% 400/400 [03:02<00:00, 2.18it/s]



- By permutation importance method

```
!pip install eli5
```

```
Collecting eli5
  Downloading eli5-0.11.0-py2.py3-none-any.whl (106 kB)
     |████████████████████████████████| 106 kB 18.8 MB/s
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from eli5) (1.4.1)
Requirement already satisfied: attrs>16.0.0 in /usr/local/lib/python3.7/dist-packages (from eli5) (21.4.0)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from eli5) (1.15.0)
Requirement already satisfied: numpy>=1.9.0 in /usr/local/lib/python3.7/dist-packages (from eli5) (1.21.6)
Requirement already satisfied: scikit-learn>=0.20 in /usr/local/lib/python3.7/dist-packages (from eli5) (1.0.2)
Requirement already satisfied: graphviz in /usr/local/lib/python3.7/dist-packages (from eli5) (0.10.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-packages (from eli5) (2.11.3)
Requirement already satisfied: tabulate>=0.7.7 in /usr/local/lib/python3.7/dist-packages (from eli5) (0.8.9)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn>=0.20->eli5) (3.1.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn>=0.20->eli5) (1.1.0)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-packages (from jinja2->eli5) (2.0.1)
Installing collected packages: eli5
Successfully installed eli5-0.11.0
```

```
from keras.wrappers.scikit_learn import KerasClassifier, KerasRegressor
import eli5
from eli5.sklearn import PermutationImportance

perm = PermutationImportance(estimator, random_state=1).fit(X_train,y_train)
eli5.show_weights(perm, feature_names = X_train.columns.tolist())
```

Weight	Feature
1.9703 ± 0.0788	Total Charges
0.7309 ± 0.0430	Operating Certificate Number
0.2972 ± 0.0342	CCS Procedure Code
0.2778 ± 0.0551	APR DRG Code
0.2682 ± 0.0141	APR MDC Code
0.2586 ± 0.0242	Length of Stay
[illegible]	[illegible]
[illegible]	[illegible]rtment Indicator
0.0522 ± 0.0192	Payment Typology 3
0.0448 ± 0.0123	CCS Diagnosis Code
0.0329 ± 0.0060	APR Severity of Illness Code
0.0155 ± 0.0037	Sur
0.0099 ± 0.0028	Payment Typology 2
0.0091 ± 0.0022	Home or Self Care
0.0079 ± 0.0007	APR Medical Surgical Description
0.0072 ± 0.0013	Gender
0.0057 ± 0.0022	Race
0.0051 ± 0.0023	Min
0.0037 ± 0.0017	Age Range
0.0018 ± 0.0007	Ethnicity
	… 9 more …

- Partial Dependence Plot

```
!pip install pdpbox
```

```
Collecting pdpbox
  Downloading PDPbox-0.2.1.tar.gz (34.0 MB)
     |████████████████████████████████| 34.0 MB 5.1 MB/s
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from pdpbox) (1.3.5)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from pdpbox) (1.21.6)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from pdpbox) (1.4.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from pdpbox) (1.1.0)
Requirement already satisfied: psutil in /usr/local/lib/python3.7/dist-packages (from pdpbox) (5.4.8)
Collecting matplotlib==3.1.1
  Downloading matplotlib-3.1.1-cp37-cp37m-manylinux1_x86_64.whl (13.1 MB)
     |████████████████████████████████| 13.1 MB 18.1 MB/s
Requirement already satisfied: sklearn in /usr/local/lib/python3.7/dist-packages (from pdpbox) (0.0)
```

```
features=['Operating Certificate Number','Length of Stay',
'CCS Diagnosis Code',
'CCS Procedure Code',
'APR DRG Code',
'APR MDC Code',
'APR Severity of Illness Code',
'Total Charges',
'Gender',
'Age Range' ,
'Race',
'Disposition',
'Admission Type',
'Emergency Department Indicator',
'Abortion Edit Indicator',
'Ethnicity',
'APR Severity of Illness Description',
'APR Risk of Mortality',
'APR Medical Surgical Description',
'Payment Typology 1',
'Payment Typology 2',
'Payment Typology 3',
'Home or Self Care',
'Extreme',
'Minor',
```



```
'Ext',
'Min']


from pdpbox import pdp, get_dataset, info_plots

# Gather pdp data
pdp_los = pdp.pdp_isolate(model = estimator,
                                dataset = X_train,
                                model_features = features,
                                feature='Length of Stay')


pdp.pdp_plot(pdp_los, 'Length of Stay',
             x_quantile=False,
            plot_pts_dist=False)
plt.show()
```

```
findfont: Font family ['Arial'] not found. Falling back to DejaVu Sans.
findfont: Font family ['Arial'] not found. Falling back to DejaVu Sans.
findfont: Font family ['Arial'] not found. Falling back to DejaVu Sans.
findfont: Font family ['Arial'] not found. Falling back to DejaVu Sans.
```

Using the significant features from that we learnt in neural networks to use on random forest

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

# To ignore any warnings
import warnings
warnings.filterwarnings("ignore")
```

```
df = pd.read_csv(r'/content/drive/MyDrive/RF+NN/dataset.csv')
```

```
del df['Unnamed: 0']
```

Saved successfully!

```
rating Certificate Number', 'APR DRG Code', 'Length of Stay', 'CCS Procedure Code', 'APR MDC Code', 'Med'
```

```
df2=df[finalfeatures]
```

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.inspection import permutation_importance
from matplotlib import pyplot as plt
```

```
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
df=df.dropna()
X = df2
y = df2.pop('Total Costs')
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.50, random_state=40)
```

```
rf = RandomForestRegressor(n_estimators=35, depth=50)
```

```
rf.fit(X_train, y_train)
```

```
RandomForestRegressor(n_estimators=35)
```

```
y_pred=rf.predict(X_test)
```

```
from sklearn.metrics import r2_score
r2=r2_score(y_test, y_pred)
```

```
print(r2)
```

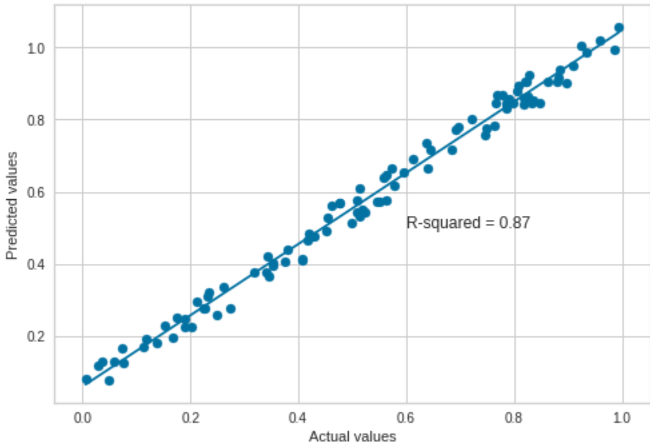
```
0.8731715401839102
```

```
y_test = np.random.rand(100) # Random Data
y_pred = y_test + np.random.rand(100)*0.1 # Random Data
```

```
r_squared = 0.87
plt.scatter(y_test,y_pred)
plt.xlabel('Actual values')
plt.ylabel('Predicted values')

plt.plot(np.unique(y_test), np.poly1d(np.polyfit(y_test, y_pred, 1))(np.unique(y_test)))

plt.text(0.6, 0.5, 'R-squared = %0.2f' % r_squared)
plt.show()
```



Thus this deep hybrid learning model has achieved a accuracy rate of 87% which is much better than random forest and neural network alone separately

Saved successfully!