Logistic Regression on Hospital Dataset

Master of Science in Data Analytics Capstone: Data Analytics Report and Executive Summary

Brandi Enrietti, Graduate Student

Student ID 924116

Western Governors University

Table of Contents

Executive Summary

This report is a written review of the data analysis conducted with logistic regression as a requirement of Western Governors Master of Science Data Analytics Capstone. The hypothesis for this analysis is:

H0: A predictive logistic regression model cannot be made from the Hospital Charges in High-Risk vs. Not High-Risk Patients who are admitted dataset.

H1: A predictive logistic regression model can be constructed from the Hospital Charges in High-Risk vs. Not High-Risk Patients who are admitted dataset.

The data was found to have a distribution of labels of 0.7 and 0.3. The first model showed an accuracy of 72% with 56% precision and a 14% recall. The data was limited by the distribution of the data. It is recommended that a more distributed dataset be obtained before running the model again.

Logistic Regression on Hospital Dataset

Research Question

This report is a written review of the data analysis conducted with logistic regression as a requirement of Western Governors Master of Science Data Analytics Capstone. The hypothesis for this analysis is:

H_0: A predictive logistic regression model cannot be made from the Hospital Charges in High-Risk vs. Not High-Risk Patients who are admitted dataset.

H_1: A predictive logistic regression model can be constructed from the Hospital Charges in High-Risk vs. Not High-Risk Patients who are admitted dataset.

Richard L. Fuller, Richard F. Averill, John Muldoon, and John S. Hughes published an article in the National Library of Medicine in 2016 that looked at clinical risk-adjustment the ability to standardize the comparison of individuals with different health needs is based upon 2 main alternative approaches: regression models and clinical categorical models. Their study utilized regression models to analyze variables of interest such as cost, complications, readmissions, or mortality, (Fuller, R. L., Averill, R. F., Muldoon, J. H., & Hughes, J. S. 2016).

The contribution of this study to the field of Data Analytics and the MSDA program is to create a reusable predictive model of logistic regression which can evaluate the relationship between diagnosis, hospital charges, number of discharges, and Medicare payments in the given dataset. To summarize, the logistic regression model supports a hypothesis that examines the given variables ability to predict outcomes. The variables will be defined, and the statistical framework will be specified. The model will test the association of the predictor variables and accuracy will be determined (Fuller, R. L., Averill, R. F., Muldoon, J. H., & Hughes, J. S. 2016).

Data Collection

The data collected for this analysis is publicly available information and was provided by Data.gov The original data set obtained contained 163,065 rows and 13 columns.

The dataset is made available to Kaggle through Data.Gov. The data set includes the following variables: “diagnosis”, “high_risk” “provider_id”, “facility_name”, “facility_address”, “facility_city”, “facility_state”, “facility_zip”, “region”, “total_discharges”, “total_charge”, “average_total_payments”, and “average_medicare_payments”.

https://www.kaggle.com/datasets/speedoheck/inpatient-hospital-charges?select=inpatientCharges.csv

The predictor variables are broken down as follows:

Field	Type
Diagnosis	Categorical
High_Risk	Object
Provider_ID	Continuous
Facility_Zip	Continuous
Region	Categorical
Average_Medicare_Payments	Categorical
Average_Total_Payments	Continuous
Total_Discharges	Continuous

There is no information that would make the hospitals associated with this analysis identifiable.

The dataset is limited to the total charge of the inpatient visit of each case. When looking at this data from a business perspective that information must be carefully considered in conjunction with this analysis when used to make any future predictions or determinations. There are no delimitations to this study.

Data Extraction and Preparation

The data set was evaluated for data integrity by evaluating both quantitative and qualitative variables for null and duplicate values. This was done as part of the pre-process of the raw data so that it is easily and accurately analyzed, (DataCamp, nd). No null or duplicates were found in the dataset, had there been any, the solution would have been to impute or remove any missing variables and drop all duplicates. All categorical variables were converted to binary variables with the creation of dummy variables. The overall data sparsity percentage is less than 2%.

This analysis was conducted with Python. Python when compared to R for running predictive modeling has been found time and time again to be comparable and many of the differences in the systems come down to personal preference. However, visibility of the data is more advanced in Python (Kan, 2018). Comparison of Python to SAS for this type of analysis shows that SAS is more costly and while it is suitable for complex statistical operations it lacks data visualization and machine learning techniques, (TechVidVan, 2021). In conclusion, Python is the preferred language for data analysis and the packages available greatly support a sophisticated and yet easy method of examing the data set for this hypothesis, (cbtnuggets, 2020).

Analysis

Logistic regression is performed on the dataset utilizing Jupyter Notebook, Pandas and sklearn. The dependent variable is “High_Risk” while the predictor variables are “total_discharges”, “total_charge”, and “average_medicare_payments”. The data is visualized utilizing bivariate and univariate graphs to identify the variables that have a linear relationship.

The dependent and predictor variables identified above will be grouped together and separated to (x) features and (y) target, (Brownlee, 2022)

To build the logistic predictive model a train test split is required and was performed at an 75/25 split. Feature scaling with sklearn and StandardScaler was performed for normalization. The model was trained utilizing sklearn.linear_model which gives access to LinearRegression, (Shilash, 2022).

This process involves fitting the model to learn the coefficients that will be interpreted next. The last step before evaluating the linear model was to make predictions based on the model. Finally, once the error metrics were obtained the model was used for feature selection. According to Massaron and Boschetti (2016) this process involves removing the features that have minimal impacts on (y) in hopes of increasing the accuracy of the model. This process is performed until an 80% or higher accuracy model is obtained.

```
#Logistic Regression Analysis
```

```
#Import Libraries and Packages
import pandas as pd
from pandas import Series
from pandas import DataFrame

import numpy as np
from numpy import ndarray
from numpy.random import randn
from numpy import loadtxt, where

import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
from matplotlib.patches import Patch
from matplotlib.lines import Line2D
plt.rc("font", size=14)
get_ipython().run_line_magic('matplotlib', 'inline')

import seaborn as sns

import random

import scipy

from tkinter import *

from scipy import stats

from statsmodels.stats.outliers_influence import variance_inflation_factor
import statsmodels.api as sm
import statsmodels.formula.api as smf

from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn import preprocessing
from sklearn import datasets
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.metrics import mean_squared_error
from sklearn.metrics import classification_report
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.datasets import make_classification
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVR
from sklearn.metrics import log_loss, roc_auc_score, recall_score, precision_score, average_precision_score, f1_score, classi
from sklearn.datasets import load_digits


from IPython.core.display import HTML
from IPython.display import display

import os

import pylab
from pylab import scatter, show, legend, xlabel, ylabel
```

```
#Version data
print("pandas version " + pd.__version__)
print("numpy version " + np.__version__)
print("scipy version " + scipy.__version__)
print("matplotlib version " + matplotlib.__version__)
print("seaborn version " + sns.__version__)
print("statsmodels version " + sm.__version__)
```

```
pandas version 1.4.2
numpy version 1.21.5
scipy version 1.7.3
matplotlib version 3.5.1
seaborn version 0.11.2
statsmodels version 0.13.2
```

```
#Display adjustment
#pd.options.display.float_format = '{:,.6f}'.format
pd.set_option('display.max_columns', None) #show all columns
```

```
#Ignore future warning code
import warnings
warnings.filterwarnings('ignore')
```

```
#Read csv
df = pd.read_csv('inpatientcharges.csv')
```

```
#view data layout
df.head()
```

	diagnosis	high_risk	provider_id	facility_name	facility_address	facility_city	facility_state	facility_zip	region	total_discharges	total_charg
0	039 - EXTRACRANIAL PROCEDURES W/O CC/MCC	No	10001	SOUTHEAST ALABAMA MEDICAL CENTER	1108 ROSS CLARK CIRCLE	DOTHAN	AL	36301	AL - Dothan	91	32963.0
1	039 - EXTRACRANIAL PROCEDURES W/O CC/MCC	No	10005	MARSHALL MEDICAL CENTER SOUTH	2505 U S HIGHWAY 431 NORTH	BOAZ	AL	35957	AL - Birmingham	14	15131.8
2	039 - EXTRACRANIAL PROCEDURES W/O CC/MCC	No	10006	ELIZA COFFEE MEMORIAL HOSPITAL	205 MARENGO STREET	FLORENCE	AL	35631	AL - Birmingham	24	37560.3
3	039 - EXTRACRANIAL PROCEDURES W/O CC/MCC	No	10011	ST VINCENT'S EAST	50 MEDICAL PARK EAST DRIVE	BIRMINGHAM	AL	35235	AL - Birmingham	25	13998.2
4	039 - EXTRACRANIAL PROCEDURES W/O CC/MCC	No	10016	SHELBY BAPTIST MEDICAL CENTER	1000 FIRST STREET NORTH	ALABASTER	AL	35007	AL - Birmingham	18	31633.2

```
#Number of columns and rows
df.shape
```

```
(163065, 13)
```

```
#Check data for null values
df.isnull().sum()
```

```
diagnosis                   0
high_risk                   0
provider_id                 0
facility_name               0
facility_address            0
facility_city               0
facility_state              0
facility_zip                0
region                      0
total_discharges            0
total_charge                0
average_total_payments      0
average_medicare_payments   0
dtype: int64
```

```
#Check data for duplicates
df.duplicated().sum()
```

```
0
```

```
#Index, Datatype
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 163065 entries, 0 to 163064
Data columns (total 13 columns):
 #   Column                     Non-Null Count   Dtype
---  ------                     --------------   -----
 0   diagnosis                  163065 non-null  object
 1   high_risk                  163065 non-null  object
 2   provider_id                163065 non-null  int64
 3   facility_name              163065 non-null  object
 4   facility_address           163065 non-null  object
 5   facility_city              163065 non-null  object
 6   facility_state             163065 non-null  object
 7   facility_zip               163065 non-null  int64
 8   region                     163065 non-null  object
 9   total_discharges           163065 non-null  int64
 10  total_charge               163065 non-null  float64
 11  average_total_payments     163065 non-null  float64
 12  average_medicare_payments  163065 non-null  float64
dtypes: float64(3), int64(3), object(7)
memory usage: 16.2+ MB
```

```
#Drop provider_id column as it is just a unique id
df.drop("provider_id",inplace=True,axis=1)
```

```
#Count # of variables in relationship to cardiac disease or no cardiac disease
df.groupby('high_risk').count()
```

	diagnosis	facility_name	facility_address	facility_city	facility_state	facility_zip	region	total_discharges	total_charge	average_total_payments
high_risk										
No	116081	116081	116081	116081	116081	116081	116081	116081	116081	116081
yes	46984	46984	46984	46984	46984	46984	46984	46984	46984	46984

```
print(df.total_charge.max())
```

929118.9

```
#Convert Categorical variable to Continous variable with dummy function
df['dummy_high_risk'] = [1 if v == 'yes' else 0 for v in df['high_risk']]
```

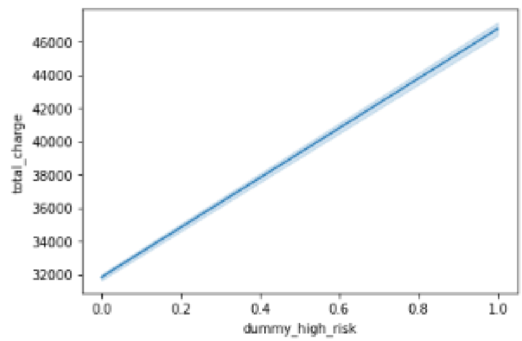
```
#Drop y/n column
df = df.drop(columns=['high_risk'])
```

```
#Summary statistics
df.describe()
```

	facility_zip	total_discharges	total_charge	average_total_payments	average_medicare_payments	dummy_high_risk
count	163065.000000	163065.000000	163065.000000	163065.000000	163065.000000	163065.000000
mean	47938.121908	42.776304	36133.954224	9707.473804	8494.490964	0.288131
std	27854.323080	51.104042	35065.365931	7664.642598	7309.467261	0.452894
min	1040.000000	11.000000	2459.400000	2673.000000	1148.900000	0.000000
25%	27261.000000	17.000000	15947.160000	5234.500000	4192.350000	0.000000
50%	44309.000000	27.000000	25245.820000	7214.100000	6158.460000	0.000000
75%	72901.000000	49.000000	43232.590000	11286.400000	10056.880000	1.000000
max	99835.000000	3383.000000	929118.900000	156158.180000	154620.810000	1.000000

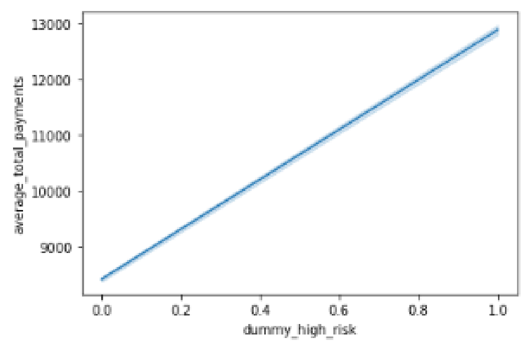
```
#Lineplot
sns.lineplot(x= "dummy_high_risk", y= "total_charge", data=df)
```

```
<AxesSubplot:xlabel='dummy_high_risk', ylabel='total_charge'>
```



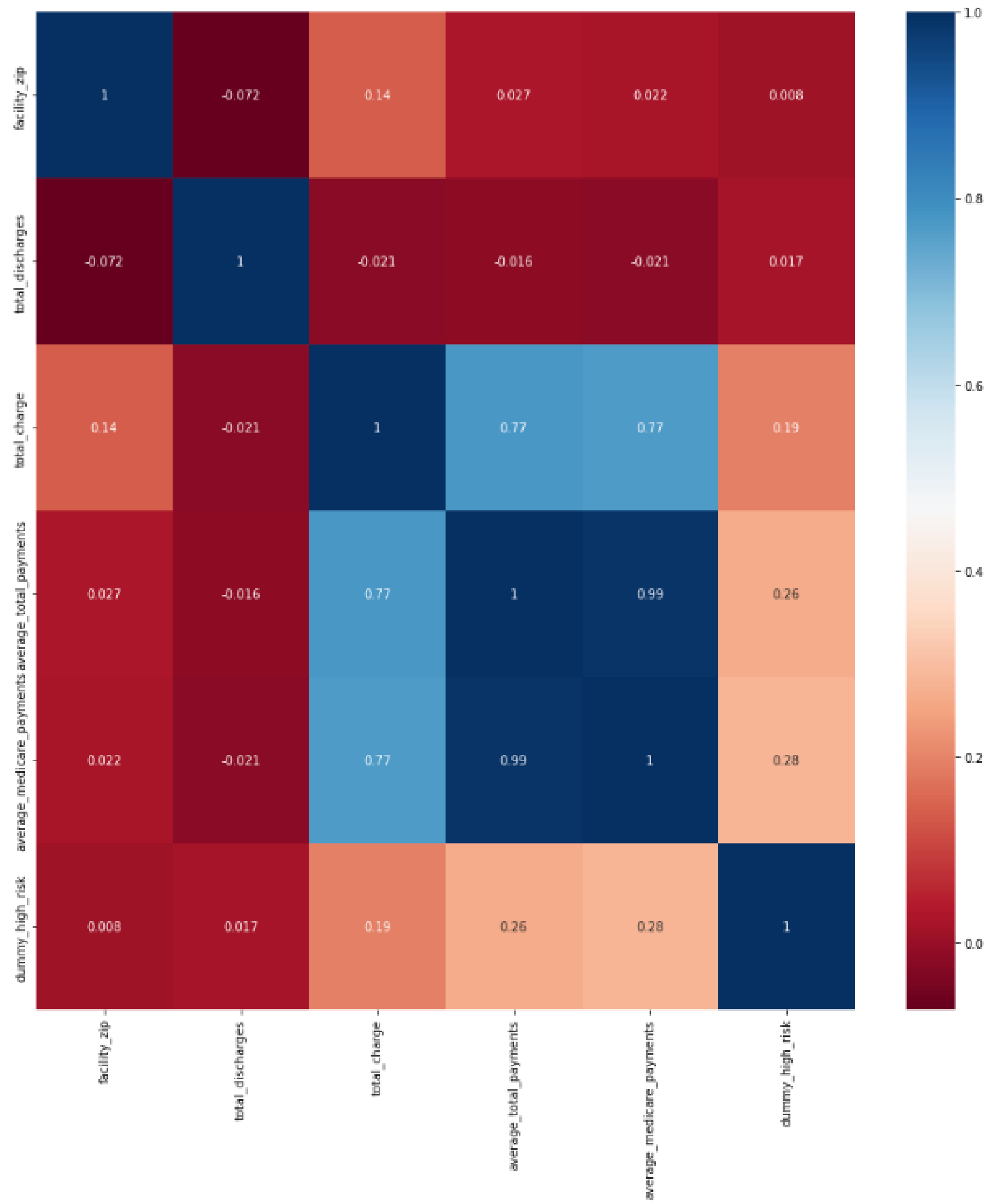

```
#Lineplot
sns.lineplot(x= "dummy_high_risk", y= "average_total_payments", data=df)
```

```
<AxesSubplot:xlabel='dummy_high_risk', ylabel='average_total_payments'>
```



```
#Correlation of entire dataset
corr = df.corr()
fig, ax = plt.subplots(figsize=(15,15))
sns.heatmap(corr, xticklabels = corr.columns, yticklabels = corr.columns, cmap='RdBu', annot=True)
```

<AxesSubplot:>



Data Summary and Implications

In conclusion, the initial model was found to have a distribution of labels of 0.7 and 0.3. The first model showed an accuracy of 72% with 56% precision and a 14% recall. The data was limited by the distribution of the data. It is recommended that a more distributed dataset be obtained before running the model again. The reduced model with “total_charge”, “average_total_payments”, and “total_discharges” showed an accuracy score of 71% with an AUC score of 75%. Based on these results the hypothesis that a predictive logistic regression model cannot be made from the Hospital Charges in High-Risk vs. Not High-Risk Patients who are admitted dataset is accepted. Further distribution of data is required for acceptance of the null hypothesis, (Thorpe, 1988).

```
#Convert columns to a list
df.columns.tolist()
```

```
['diagnosis',
 'facility_name',
 'facility_address',
 'facility_city',
 'facility_state',
 'facility_zip',
 'region',
 'total_discharges',
 'total_charge',
 'average_total_payments',
 'average_medicare_payments',
 'dummy_high_risk']
```

```
#Initial model
```

```
#Split dataset in features and target variable
feature_cols = ['total_charge', 'average_medicare_payments', 'average_total_payments', 'total_discharges']
X = df[feature_cols] # Features
y = df.dummy_high_risk # Target variable
```

```
#Split X and y into training and testing sets
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=0)
```

```
#Feature scaling(normalizing)
scale=StandardScaler()
X_train = scale.fit_transform(X_train)
X_test = scale.transform(X_test)
```

```
#check for distribution of labels
y_train.value_counts(normalize=True)
```

```
0    0.711238
1    0.288762
Name: dummy_high_risk, dtype: float64
```

```
#Instantiate the model (using the default parameters)
logreg = LogisticRegression()

#Fit the model with data
logreg.fit(X_train,y_train)
y_pred=logreg.predict(X_test)
print(logreg.score(X_test, y_test))
```

```
0.7227904923099566
```

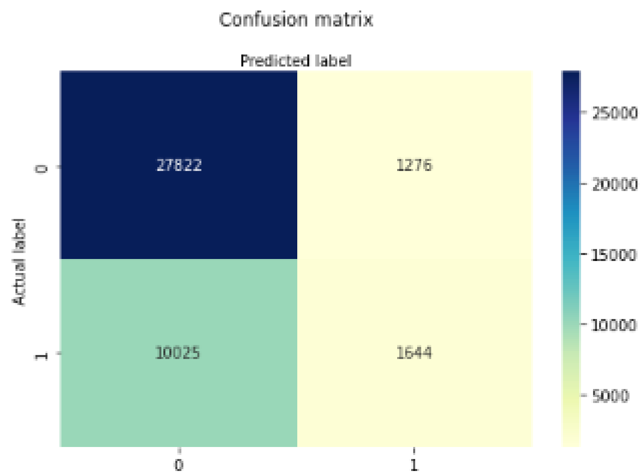
```
#Confusion matrix
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

```
array([[27822,  1276],
       [10025,  1644]], dtype=int64)
```

```
#Classes
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)

#Create heatmap to visualize results of the model
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

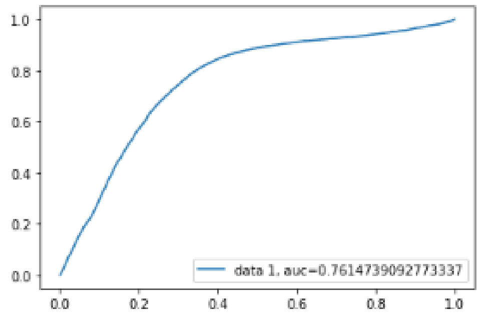
```
Text(0.5, 257.44, 'Predicted label')
```




```
#Accuracy, Precision, Recall
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("Precision:",metrics.precision_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
```

```
Accuracy: 0.7227904923099566
Precision: 0.563013698630137
Recall: 0.1408861084925872
```

```
#AUC/ROC Curve
y_pred_proba = logreg.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```




```
#Reduced Model
```

```
#split dataset in features and target variable
feature_cols = ['total_charge', 'average_total_payments', 'total_discharges']
X = df[feature_cols] # Features
y = df.dummy_high_risk # Target variable
```

```
#Split X and y into training and testing sets
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=0)
```

```
#Feature scaling
scale=StandardScaler()
X_train = scale.fit_transform(X_train)
X_test = scale.transform(X_test)
```

```
#check for distribution of labels
y_train.value_counts(normalize=True)
```

```
0    0.711238
1    0.288762
Name: dummy_high_risk, dtype: float64
```

```
#Instantiate the model (using the default parameters)
Rlogreg = LogisticRegression()

#Fit the model with data
logreg.fit(X_train,y_train)
y_pred=logreg.predict(X_test)
print(logreg.score(X_test, y_test))
```

```
0.7141315279515295
```

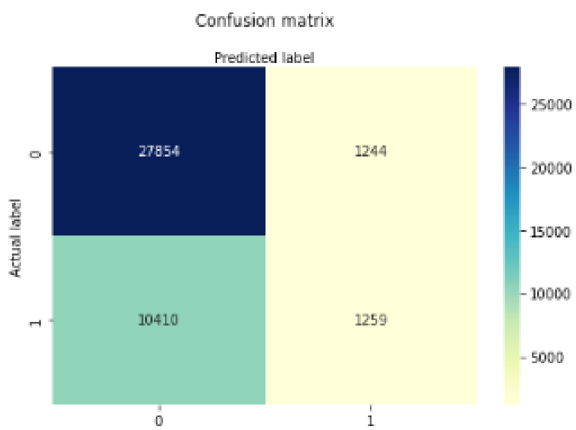
```
#Confusion matrix
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

```
array([[27854,  1244],
       [10410,  1259]], dtype=int64)
```

```
#Classes
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)

#Create heatmap to visualize results of the model
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

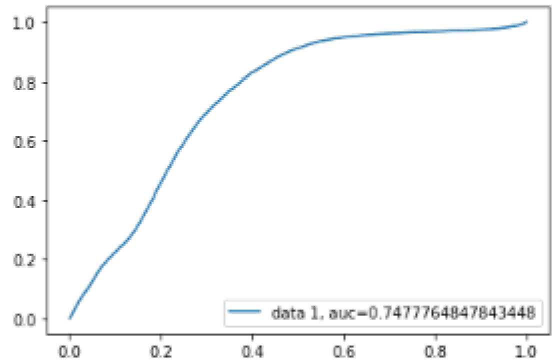
```
Text(0.5, 257.44, 'Predicted label')
```




```
#Accuracy, Precision, Recall
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("Precision:",metrics.precision_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
```

```
Accuracy: 0.7141315279515295
Precision: 0.5029964043148222
Recall: 0.10789270717285114
```

```
#AUC/ROC Curve
y_pred_proba = logreg.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```



```
#Final model
```

```
features = ['total_charge',]
target = 'dummy_high_risk'
m = len(X)
```

```
X = df[features]
y = df[target]
model = sm.OLS(y, X).fit()
predictions = model.predict(X)

model.summary()
```

OLS Regression Results

Dep. Variable:	dummy_high_risk	R-squared (uncentered):	0.249
Model:	OLS	Adj. R-squared (uncentered):	0.249
Method:	Least Squares	F-statistic:	5.394e+04
Date:	Wed, 01 Jun 2022	Prob (F-statistic):	0.00
Time:	15:55:10	Log-Likelihood:	-1.0663e+05
No. Observations:	163065	AIC:	2.133e+05
Df Residuals:	163064	BIC:	2.133e+05
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
total_charge	5.315e-06	2.29e-08	232.248	0.000	5.27e-06	5.36e-06

Omnibus:	7827.774	Durbin-Watson:	0.098
Prob(Omnibus):	0.000	Jarque-Bera (JB):	9012.981
Skew:	0.575	Prob(JB):	0.00
Kurtosis:	3.072	Cond. No.	1.00

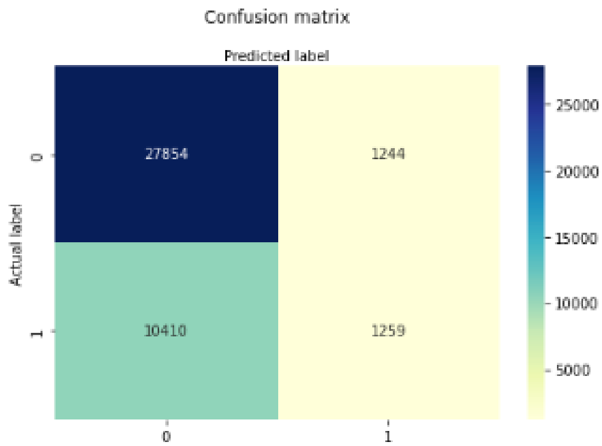
Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```
#Confusion matrix
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

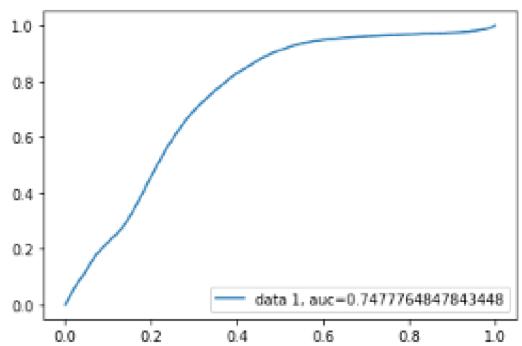
```
array([[27854,  1244],
       [10410,  1259]], dtype=int64)
```

```
#Classes
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
#Create heatmap to visualize results of the model
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

```
Text(0.5, 257.44, 'Predicted label')
```




```
#AUC/ROC Curve
y_pred_proba = logreg.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```



References

Aryal, P. (2016, September 19). *Hospital charges for inpatients*. Kaggle. Retrieved May 28, 2022, from https://www.kaggle.com/datasets/speedoheck/inpatient-hospital-charges?select=inpatientCharges.csv

Brownlee, J. (2020, August 14). *Linear regression for machine learning*. Machine Learning Mastery. Retrieved May 28, 2022, from https://machinelearningmastery.com/linear-regression-for-machine-learning/

CBTNuggets. (2018, September 20). Why Data Scientists Love Python.

https://www.cbtnuggets.com/blog/technology/data/why-data-scientists-love-python

Massaron, L. & Boschetti, A. (2016). Regression Analysis with Python. Packt Publishing.

DataCamp. (n.d.). *Data preparation with Pandas*. Retrieved May 29, 2022, from https://www.datacamp.com/tutorial/data-preparation-with-pandas

Fuller, R. L., Averill, R. F., Muldoon, J. H., & Hughes, J. S. (2016). Comparison of the Properties of Regression and Categorical Risk-Adjustment Models. The Journal of ambulatory care management, 39(2), 157–165. https://doi.org/10.1097/JAC.0000000000000135

Kan, E. (2018, December 10). *Data science 101: Is python better than R?* Medium. Retrieved May 29, 2022, from https://towardsdatascience.com/data-science-101-is-python-better-than-r-b8f258f57b0f

Linear Regression - Project Exercise. (n.d.). Amete.github.io. Retrieved April 27, 2022, from https://amete.github.io/DataSciencePortfolio/Udemy/Python-DS-and-ML-Bootcamp/Linear_Regression_Project.html

Mawardi, D. (2017, August 22). *Linear regression in python*. Medium. Retrieved May 28, 2022, from https://towardsdatascience.com/linear-regression-in-python-9a1f5f000606

SAS vs R vs python - the battle for data science! TechVidvan. (2021, July 6). Retrieved May 29, 2022, from https://techvidvan.com/tutorials/sas-vs-r-vs-python/

Shilash, M. (2022, March 22). *Linear regression with python implementation*. Analytics Vidhya. Retrieved May 28, 2022, from https://www.analyticsvidhya.com/blog/2022/02/linear-regression-with-python-implementation/

Sklearn.linear_model.linearregression. scikit. (n.d.). Retrieved May 29, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

Thorpe, K. E. (1988). The Use of Regression Analysis to Determine Hospital Payment: The Case of Medicare's Indirect Teaching Adjustment. *Inquiry*, *25*(2), 219–231. http://www.jstor.org/stable/29771954