Diabetes 130-US Hospitals Dataset

Prediction using Machine Learning Models

Yogesh Jadhav, Pooja Thakoor
47969389, 94203377
University of California, Irvine

06/13/2019

1 Introduction

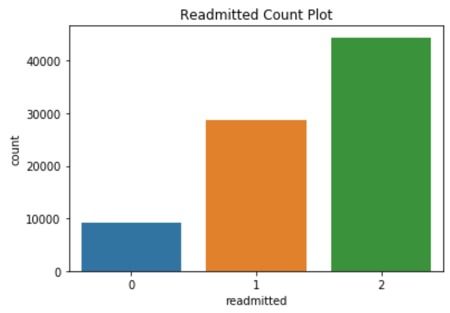
In this project, we predict hospital readmission for patients with diabetes using Diabetes 130-US Hospital Dataset. We explore the various aspects of this data by visualizing and analyzing it. We performed feature engineering and PCA to obtain data that is fit for our models to perform better. We explore various machine learning and hyperparameter tuning techniques to obtain better performance on our test data. Then we do performance evaluation of various models.

2 Dataset Overview

After performing the split operation on the dataset we observed the shape of all splits. Training data contains 82433 data points, validation data contains 9158 data points and test data contains 10175 data points. There are 49 features in the data. There are 3 class labels viz '<30', '>30' and 'no' for predicting within how many days the patient is readmitted

Data	Shape
train_features	82433 x49
train_labels	82433 x 1
validation_features	9158 x 49
validation_labels	9158 x 1
test_features	10175 x 49
test_labels	10175 x 1

2.1 Distribution of classes

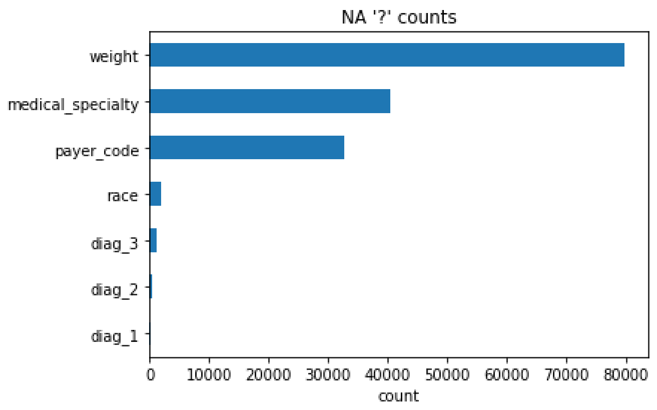


Training data is not distributed uniformly across the classes. There is a comparatively large number of data points corresponding to the class 'no' in the training data. Hence we have added *"class_weight=balanced"* to all our sklearn learners to cut some bias emerging from this unbalanced dataset.

2.1 NA '?' values in dataset

There are many features in the dataset that contains the value '?' which is an unknown value. We observed 'weight', 'medical specialty' and 'payer_code' feature has a maximum amount of unknown value. As such features are not at all correlated to the labels, we have dropped them.

Feature	Count (?)
weight	79815
medical_speciality	40500
payer_code	32558



2.2 ID Columns

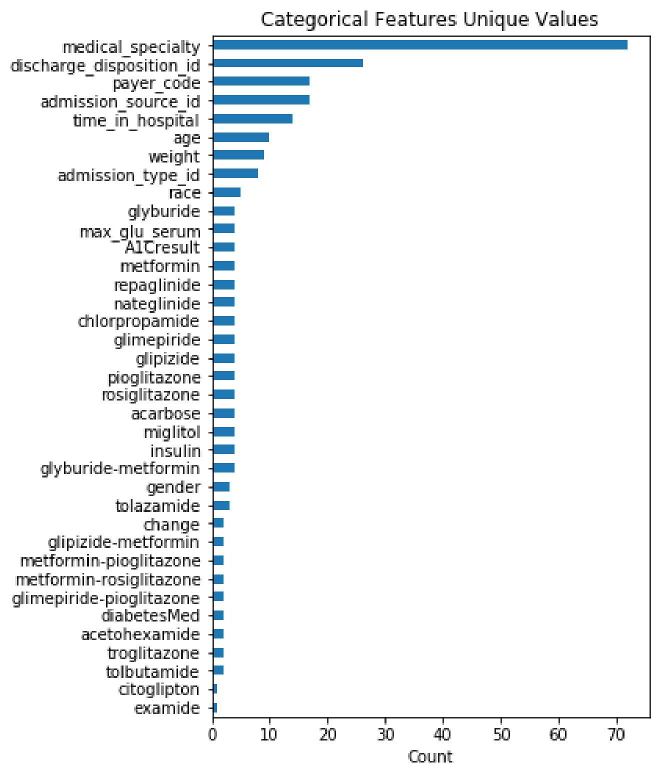
There are 2 ID columns in the dataset viz. 'encounter_id' and 'patient_nbr'. There are multiple records corresponding to a single patient in the dataset. ID features do not add any importance in our data as all the values are unique.

2.3 Categorical Features

There are around 35 categorical features in the dataset which are listed down below.

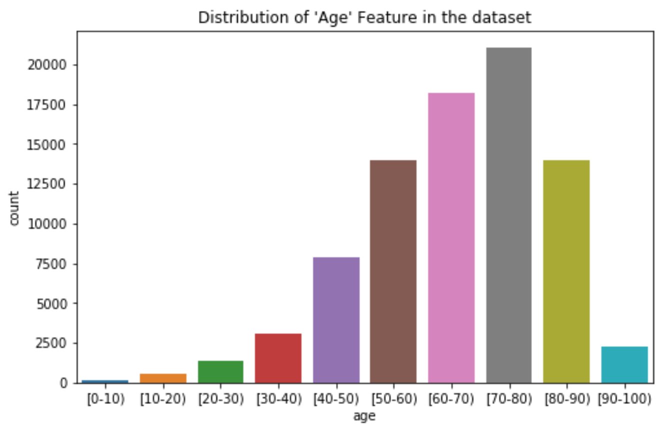
Categorical Features		
gender	miglitol	rosiglitazone

race	troglitazone	acarbose
admission_type_id	tolazamide	pioglitazone
discharge_disposition_id	insulin	glipizide
medical_specialty	glyburide-metformin	glyburide
max_glu_serum	glipizide-metformin	tolbutamide
A1Cresult	metformin-rosiglitazone	diag_1
metformin	metformin-pioglitazone	diag_2
repaglinide	change	diag_3
nateglinide	diabetesMed	acetohexamide
chlorpropamide	glimepiride	examide
citoglipton	glimepiride-pioglitazone	age, weight

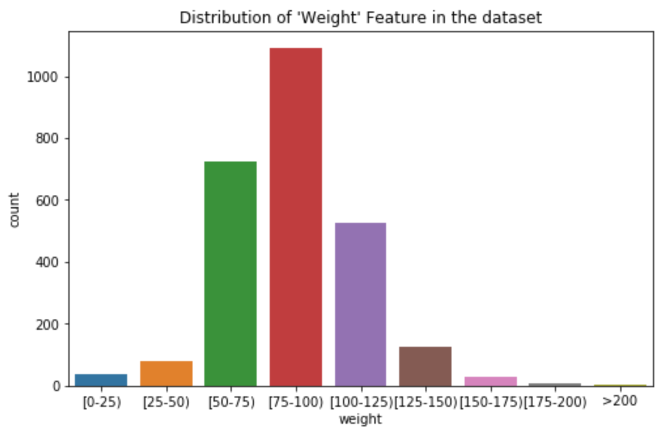


'examide', 'citoglipton' and 'glimepiride-pioglitazone' feature has a single unique value in the entire dataset. However, 'diag_x' feature has a mix of numerical and non-numerical categorical values in the dataset which are ICD9 codes. Features 'age' and 'weight' are also categorical and has values that specify a range of age and weight values.

2.4 Distribution of Age feature in the dataset



2.5 Distribution of Weight feature in the dataset



2.6 Numerical Features

The dataset contains around 9 numerical features which are listed down below.

Numerical Features		
number_inpatient	number_diagnoses	time_in_hospital
num_procedures	num_medications	number_outpatient
number_emergency		

Values corresponding to these numerical features are sparsely distributed across a range of values.

3. Preprocessing

3.1 Feature Engineering

- Drop ID Features: We dropped id columns viz. 'encounter_id', 'patient_nbr' from the dataset as these features are not important

- Drop Single Unique Value Features: Features 'examide', 'citoglipton'& 'glimepiride-pioglitazone' have single value throughout the dataset so we drop this feature

- Replace '?' with nan: Then in order to better track unknown '?' values in the dataset we replace them with python's nan value.

- Drop Single Unique Value Features: During preprocessing we dropped 'payer_code' column as had large number of nan values

- Collapse 'diag_x' Features into fewer categories: Diagnostics features viz 'diag_1', 'diag_2', 'diag_3' contains many unique values which represent ICD9 codes. In order to minimize these sub-categories, we sampled these values in the fewer categories using Wikipedia ICD9 listing into categories from -1 to 9.

- Add New Features: We added a new feature viz 'number_services' which represent the sum of the feature values in 'number_outpatient', 'number_emergency' & 'number_inpatient'

- Replace 'Unknown' Gender with nan: We replaced 'unknown' values in the gender column with nan

- Collapse 'discharge_disposition_id' features into fewer categories:

Category	Description	New Category
11	Expired	11
19	Expired at home. Medicaid only, hospice	
20	Expired in a medical facility. Medicaid only, hospice	
21	Expired, place unknown. Medicaid only, hospice	

Categories 11, 19, 20, 21 represent that the patient died afterwards. So, we collapsed these 4 categories in a single category 11.

Category	Description	New Category
18	NULL	18
25	Not Mapped	
26	Unknown/Invalid	

Categories 18, 25, 26 represent that the value is either unknown or invalid. So, we collapsed these 3 categories in a single category 18.

- Collapse 'admission_type_id' features into fewer categories

Category	Description	New Category
5	Not Available	5
6	NULL	
8	Not Mapped	

Categories 5, 6, 8 represent that the value is either unknown or invalid. So, we collapsed these 3 categories in a single category 5.

Category	Description	New Category
1	Emergency	1
2	Urgent	
7	Trauma Center	

Categories 1, 2, 7 represent that admission type is Emergency. So, we collapsed these 3 categories in a single category 1.

- Collapse 'admission_source_id' features into fewer categories:

Category	Description	New Category
9	Not Available	9
15	Not Available	
17	NULL	
20	Not Mapped	
21	Unknown/Invalid	

Categories 9, 15, 17, 20, 21 represent the unknown/null values. So, we collapsed these 5 categories in a single category 9.

- Convert 'age' and 'weight' features to numerical values:

'Age' and 'Weight' features categorical in the raw data. The values represent the range of values. So, we converted these range values to the numerical values which are average of those range values. The conversion was done as follows.

Feature	Range	Numerical Value
Age	[0-10)	5
	[10-20)	15
	[20-30)	25
	[30-40)	35
	[40-50)	45
	[50-60)	55
	[60-70)	65
	[70-80)	75

	[80-90)	85
	[90-100)	95

Feature	Range	Numerical Value
Weight	[0-25)	12.5
	[25-50)	37.5
	[50-75)	62.5
	[75-100)	87.5
	[100-120)	112.5
	[125-150)	137.5
	[150-175)	162.5
	[175-200)	187.5
	>200	212.5

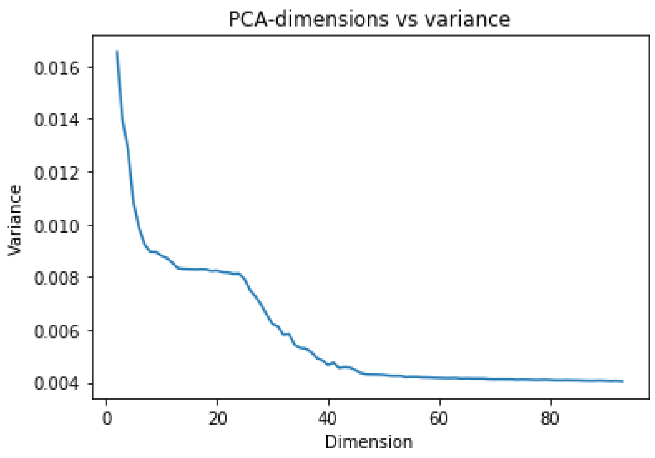
- One hot encoding of categorical data: Next important thing we do is to one hot encode all the categorical data. We use pandas utility function 'get_dummies' to one hot encode all the categorical data. We also add a new column for nan feature every time we do one hot encoding on a particular feature.

- Use Standard Scalar on the data: Once all the above processing is done we use standard scalar to rescale all our features to make each feature zero mean and unit variance.

Resulting shapes of the data after feature engineering:

Data	Shape
train_features	82433 x 284
train_labels	82433 x 1
validation_features	9158 x 284
validation_labels	9158 x 1
test_features	10175 x 284
test_labels	10175 x 1

3.2 Principle Component Analysis



The PCA-dimensions vs variances plot above clearly has two steep decreases in variances captured by two "elbow" shapes.

After the process of feature engineering, we ended up having 284 features from the initial 49 features. We decided to take the first 45 features as denoted by the steep drop in variances in between 40 and 60 dimensions.

We trained (fitted) our PCA model using the training data. We then transformed the dimensions of training, validation and test data using the model.

Resulting shapes of the data after PCA:

Data	Shape
train_features	82433 x 45
train_labels	82433 x 1
validation_features	9158 x 45
validation_labels	9158 x 1
test_features	10175 x 45
test_labels	10175 x 1

4. Model Selection And Evaluation

We used various different types of classification techniques for this problem. LogisticRegression and LinearSVM being the linear classifiers, DecisionTree Classifier, Ensemble learners like RandomForest Classifiers.

4.1 Baseline Models

- Random Forest
- Decision Tree
- SVM with rbf kernel
- Logistic Regression

- Neural Network (MLP)

4.2 Fine-tune Hyperparameters:

We used RandomizedSearchCV for fine-tuning the hyperparameters of our models. The following steps we carried out for fine-tuning the models:

1. We created a random stratified split amounting to 25% of the training data. We used this small subset of training data as this would make tuning the parameters lot less computationally intensive for a fraction of performance decrease.

2. We strategically identified the behavior of various models on the training data such as overfitting and underfitting. We included appropriate parameters to perform RandomizedSearchCV to boost our validation score higher. We used the performance of our model on validation data as the metric in this process.

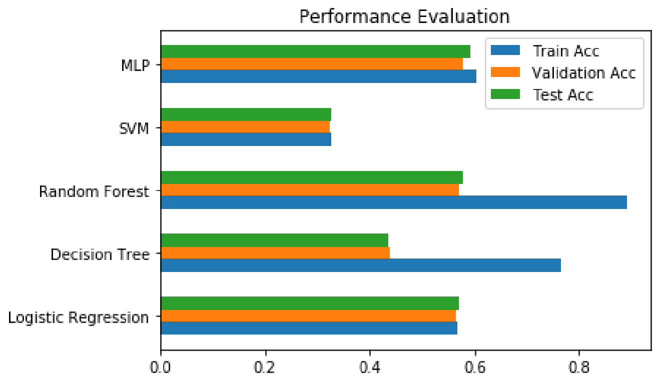
3. Once the best parameters have been identified by the process, we would use these parameters to build our learners to fit on the whole training data and to test on testing data.

5 Results

5.1 Accuracy

After finding the best hyperparameters for our baseline models using RandomizedSearchCV, we used those parameters to train our model using training data. We observed following accuracy on training, validation and test data.

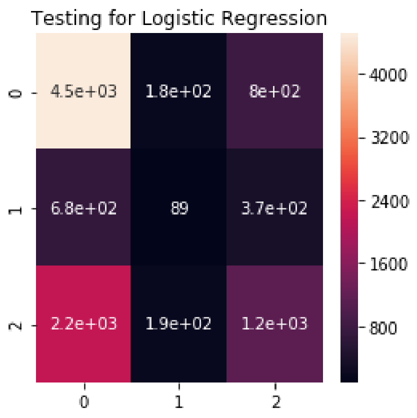
Model	Accuracy		
	Training	Validation	Test
Logistic Regression	0.57	0.57	0.57
Decision Tree	0.76	0.44	0.43
Random Forest	0.89	0.57	0.58
SVM (rbf kernel)	0.33	0.32	0.33
Neural Network	0.60	0.58	0.59



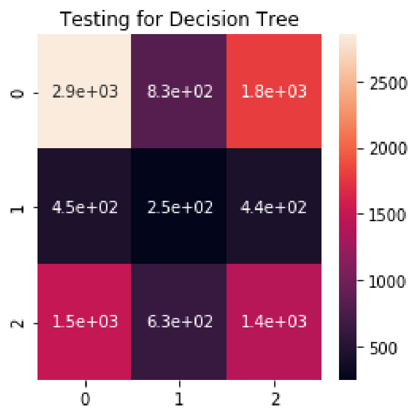
5.2 Confusion Matrix

We also observed the confusion matrices on test data for our model evaluation

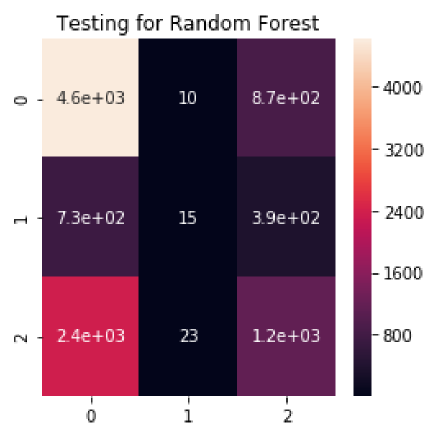
1. Logistic Regression



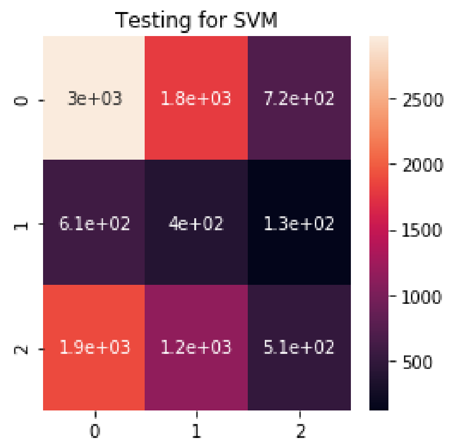
2. Decision Tree



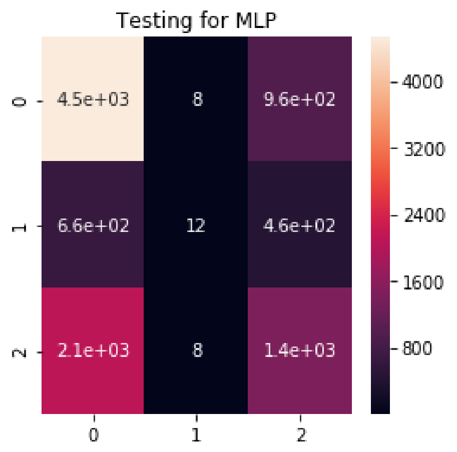
3. Random Forest



4. SVM



5. Neural Network (MLP)



6. Conclusion

We observed that MultiLayer Perceptrons are suitable for the data. We achieved the highest testing accuracy with MLP. Also, a classifier based on ensemble techniques like RandomForest classifier also gave high accuracy on test data.

7. Future Work

As data is highly skewed we plan to use SMOTE to oversample/undersample the data and see how different models perform using this new synthetic data.

We also plan to use Convolutional Neural Networks in order to predict the patient's Hospital Readmission.

9 Appendix

9.1 Division of Labour

Yogesh

Models Selection and Evaluation, Hyper-parameter tuning, Training and Validating, Principal Component Analysis, Report Work.

After binarizing the features, we got a high dimensional data of 284 features in hand. This clearly was a challenge as we had planned to permute hyperparameter values to fine-tune the model. PCA was needed for dimensionality reduction but at the same time without much loss of the information from the data. We plotted the dimensions vs variances graph and chose the best dimension where variance seems to have gotten stagnant.

We found most of the learners except RandomForest was underfitting the data. So, we chose hyper-parameters which will overcome the particular short-coming of each learner. We used RandomizedSearch for tuning the hyper-parameters.

Pooja

Data Cleaning and Processing, Data Features Engineering, Hyper-parameter tuning, Training and Validating, Report Work

I analyzed and visualized the data to understand important aspects of the data. For preprocessing I initially tried different permutations like dropping some columns/rows but later realized the importance of the data as a whole after evaluating them on baseline Models. After preprocessing, high dimensional data was consuming time for training, so, Yogesh came up with his solution of PCA. For hyperparameter tuning, we were initially using GridSearchCV but it was very time-consuming. So, I planned to use RandomizedSearchCV instead using a stratified split of training data (25% subsampling) to make the tuning faster.

9.2 Project Setup

Dependencies:

Python 3.0

Jupyter

Installing Python Dependencies:

```
pip3 install scikit-learn
pip3 install scipy
```

```
pip3 install numpy
pip3 install pandas
pip3 install seaborn
pip3 install matplotlib
pip3 install keras
```

Project code: code.ipynb

10 References

[1] https://en.wikipedia.org/wiki/List_of_ICD-9_codes

[2] https://archive.ics.uci.edu/ml/datasets/Diabetes+130-US+hospitals+for+years+1999-2008

10.1 Research papers

[1] https://www.hindawi.com/journals/bmri/2014/781670/tab1/

[2] https://www.sciencedirect.com/science/article/pii/S1877050918317873

10.2 Development Links

[1] https://towardsdatascience.com/predicting-hospital-readmission-for-patients-with-diabetes-using-scikit-learn-a2e359b15f0

[2] https://medium.com/berkeleyischool/how-to-use-machine-learning-to-predict-hospital-readmissions-part-1-bd137cbdba07

10.3 Python Libraries

[1] scikit-learn

[2] scipy

[3] numpy

[4] pandas

[5] seaborn

[6] matplotlib

[7] keras