Team Term Project Progress Report

Team 10
1. Anzal Zia Khan
2. Anowarul Kabir
Code: https://github.com/akabiraka/cs584dm_project

Project and Data Selection

To look for a possible project idea we first go through several websites and blogs such as [1], [2] and [3] and so on. Finally we choose **Diabetes 130-US hospitals** [4] dataset from UCI data repository [1]. The primary goal of this dataset is to find **Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records [5]**, where as our primary goal is to classify a patient as readmitted or not readmitted given that some diagnosis result and so on. There are several reasons for selecting this dataset. Such as:

1. The dataset is collected from 130-US hospitals and the number of individual row is 100000 where each row corresponds to one patient during encountering in the hospital, which makes this dataset as uniformly distributed.
2. The dataset is collected from real world situation which makes this dataset hard to find pattern.
3. The dataset contains many missing values which need to be cleaned. That is challenging as well as good for learning.
4. The diagnosis result for each patient is categorical which is not suitable for SVM or Deep Learning classifier. So we have to go through some data transformaion process.

Reminding these challenges, we look forward to attaining our goals by first cleaning the dataset. Before going into that we like to report some general information about the dataset which is taken from [4]. The dataset represents 10 years (1999-2008) of clinical care at 130 US hospitals and integrated delivery networks. It includes over 50 features representing patient and hospital outcomes. Information was extracted from the database for encounters that satisfied the following criteria.
(1) It is an inpatient encounter (a hospital admission).
(2) It is a diabetic encounter, that is, one during which any kind of diabetes was entered to the system as a diagnosis.
(3) The length of stay was at least 1 day and at most 14 days.
(4) Laboratory tests were performed during the encounter.
(5) Medications were administered during the encounter.
The data contains such attributes as patient number, race, gender, age, admission type, time in hospital, medical specialty of admitting physician, number of lab test performed, HbA1c test result, diagnosis, number of medication, diabetic medications, number of outpatient, inpatient, and emergency visits in the year before the hospitalization, etc.

Data Cleaning

As the dataset contains lots of missing values, we first go through cleaning process. We remove the following attributes shown in *Table 1*. *Table 1* also describes the reasons behind it. We also check if any attribute has more than 30% instances are missing, we remove that attribute.

#	**Attribute name**	**Attribute type**	**Reasons of removing**
1	encounter_id	nominal	adds no information in classification or clustering
2	patient_nbr	nominal	adds no information in classification or clustering
3	num_procedures	numerical	Corresponds to hospital procedures which has no relation with class attribute
4	weight	numerical	96.85% instances are missing
5	payer_code	nominal	39.55% instances are missing
6	medical_specialty	nominal	49.08% instances are missing
7	discharge_disposition_id	nominal	adds no information in classification or clustering
8	admission_source_id	nominal	adds no information in classification or clustering
9	admission_type_id	nominal	adds no information in classification or clustering

Table 1: Attribute removal from classification task

If the missing instance percentage of any attribute is less 30% we fill up missing values using either mode or mean based on the type of the attribute. This fill up procedure is shown on *Table 2*.

#	**Attribute name**	**Attribute type**	**Missing instances**	**Fill up process**
1	race	categorical	2.23%	mode
2	diag_1	numerical	0.02%	mean
3	diag_2	numerical	0.35%	mean
4	diag_3	numerical	1.39%	mean

Table 2: Fill up missing values of attributes

Data Description

In this chapter, we show data distribution, more specifically some attribute distribution to describe the nature of the data, class attribute values distribution, relations among attributes vs class attribute. To do that, we use data visualization technique such as histogram, scatter-plot and 3d-scatter plot. *Table 3* shows full data distribution.

Number of instances	**Number of attribute**
101766	50 (without class attribute)

Table 3: Data distribution

Figure 1 shows some histograms on attribution data distribution. Note that these attributes do not belong to diagnosis result. From here we can conclude that some attribute are so skewed, e.g. race, max_glu_serum, A1Cresult etc which makes our classification tasks more difficult in the first guess.

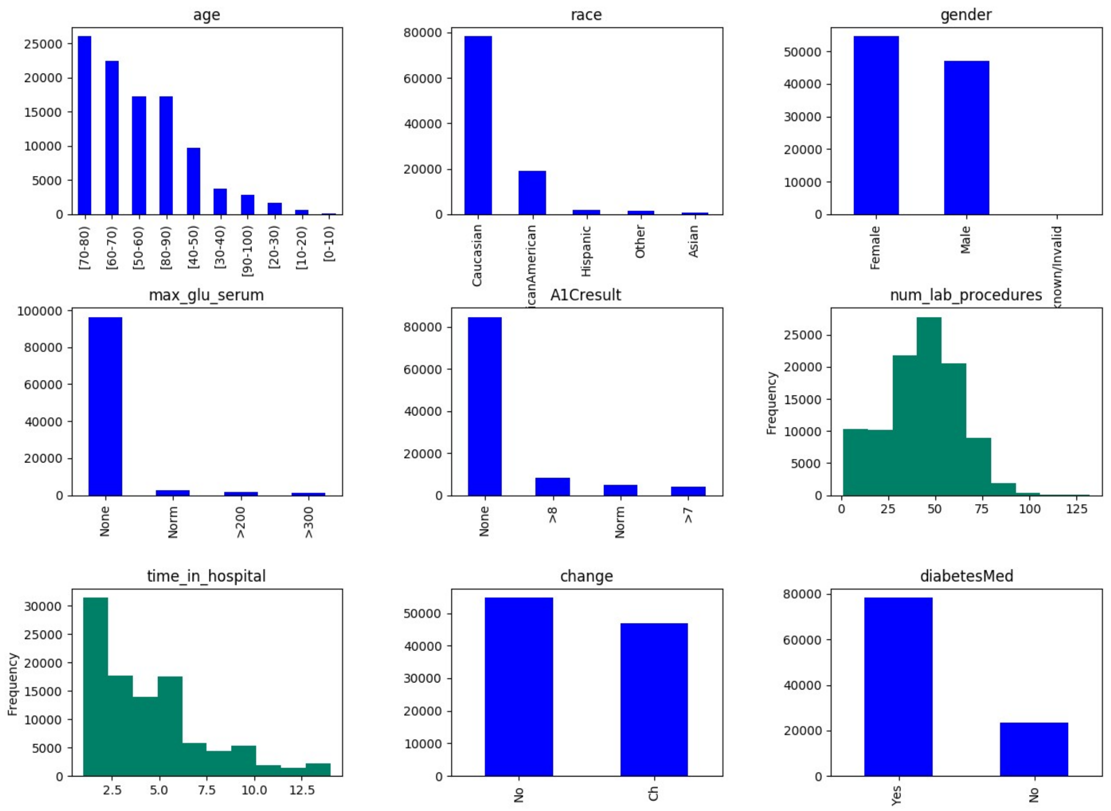


Figure 1: Data distribution (blue:categorical, green:numerical)

Class attribute: Readmitted

- NO: if the patient was not readmitted into hospital.
- <30: if the patient was readmitted into hospital within 30 days from the encountering day of that patient.
- >30: if the patient was readmitted into hospital after 30 days from the encountering day of that patient.

In Figure 2, we show the class attribute distribution. Class attribute is fairly normally distributed. Although the number of *NO* examples are far more that *<30* examples, however if we say *>30* and *<30* together represent *YES*, we can conclude it is normally distributed. Note that, we do not use *>30* and *<30* together in our work, instead they represent separate class.

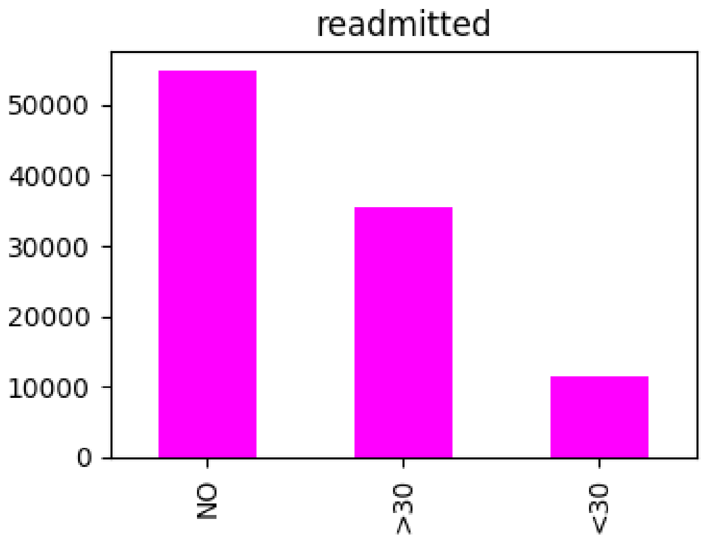


Figure 2: Class attribute distribution

We try to find if there is any relation between *num_lab_procedures* and *num_medications* because we hypothesize that if number of lab procedures increase number of medications will increase and vice versa. To prove or disprove our hypothesis we draw scatter plot. As these attributes have different range of values, so we standardize them. *Figure 3* shows scatter plot between *num_lab_procedures* and *num_medications.* From here we can not effectively say there are any relation between them.

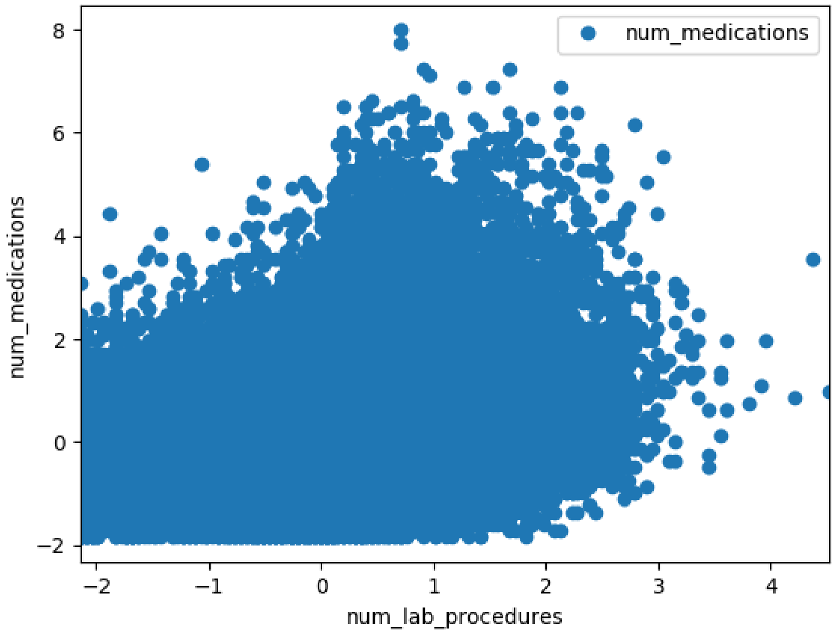


Figure 3: Scatter plot between num_lab_procedures and num_medications

We try to find relation among number of different patients (such as in_patient, out_patient, emergency_patient) vs class attribute which is shown in Figure 4, but that does not seem to be so helpful. We also plot diag_1, diag_2, diag_3 in X, Y and Z-axis vs class attribute namely *readmitted* scatter plot so that we can see diagnosis relations with class attributes as well as among themselves, *Figure 5* shows that. We assume that the number of instances is so big that visualization method is not helping so much. So from here we will directly go into classification task.

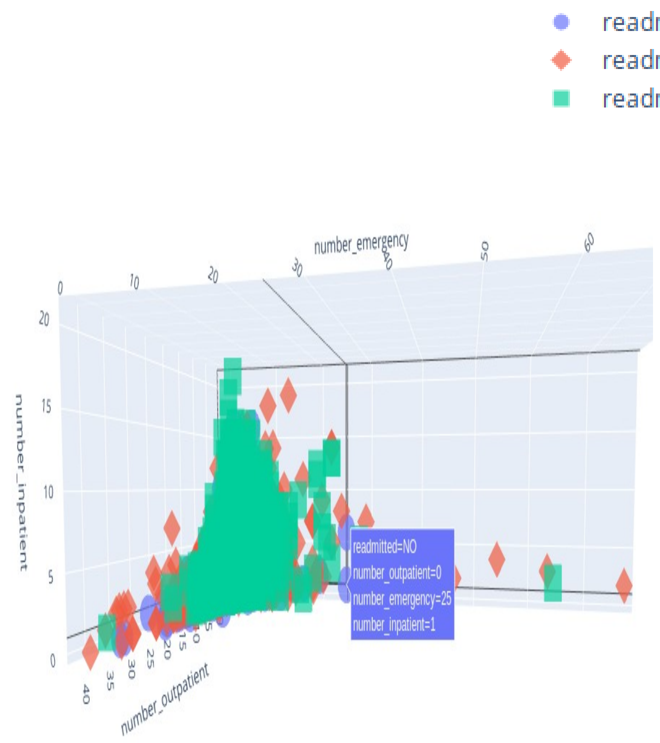


Figure 4: In, out and emergency patient number vs Readmitted(class)

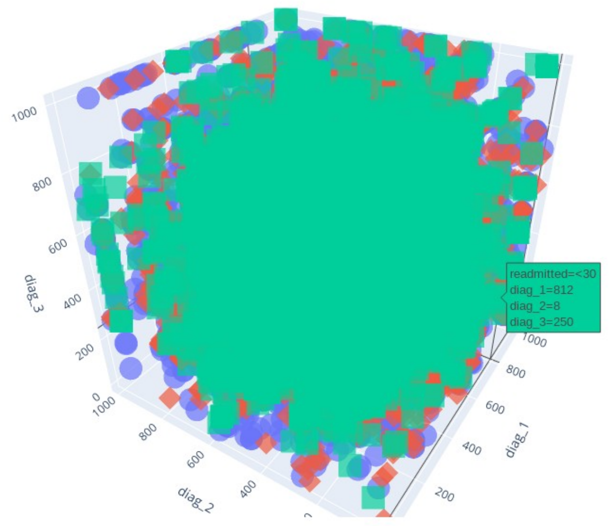


Figure 5: diag_1, diag_2, diag_3 vs Readmitted(class)

Decision Tree

a) Data Preprocessing: We have many features in our dataset which string values. We use label encoding to convert these labels into numeric form so as to make them machine-readable.

We divide the Dataset into training and test data by using the first 80000 records for training our model and the remaining 21700 records for test data.

b) 5-Fold Cross Validation: To evaluate different models we use 5-fold cross validation. We split our training set into 5 different subsets. We use 4 subsets to train our data and leave the last subset (or the last fold) as test data. We then average the model against each of the folds and then finalize our model.

We evaluate different models using 5-fold cross validation before finalizing our model.

c) Hyper Parameter Tuning: The Decision Tree has a lot of parameters that require tuning in order to derive the best possible model that reduces the generalization error as much as possible. We focused on 5 hyper parameters:

> **1) Max Depth:** Maximum number of children nodes that can grow out of a Decision Tree until the tree is cut off. If set to 3, the tree will be cut off once 3 children nodes are used.
>
> **2) Min Samples Leaf:** The minimum number of samples or data points that must be present inside the leaf node. If set to 0.04, it dictates that the tree should grow until the leaf nodes have at least 4% of the total samples in the data.
>
> **3) Splitting Criterion:** The function used to measure the quality of a split. We have two functions in the scikit library i.e. “gini” for Gini Impurity and “entropy” for Information Gain.
>
> **4) Min Samples Split:** Minimum number of samples required to split an internal node. When we increase this parameter, the tree becomes more constrained as it has to consider more samples to split a node.
>
> **5) Splitter:** The strategy used to choose the split at each node. Supported strategies are “best” to choose the best split and “random” to choose the best random split.

Hyper Parameter Tuning Method

Grid Search: A very basic hyper parameter tuning method. We build a model for each combination of all the hyper parameter values provided, evaluating each model using 5-fold cross validation and selecting the architecture which produces the best results. The scoring metric used was accuracy.

We define our hyper-parameter space as follows:

```
grid_params = {
    'criterion': ['gini','entropy'],
    'splitter': ['best','random'],
    'max_depth': [2,4,6],
    'min_samples_leaf': [0.02, 0.04],
    'min_samples_split': [0.2,0.5,0.8]
}
```

grid_object = GridSearchCV(estimator = dt, param_grid = grid_params, scoring = 'accuracy', cv = 5, n_jobs = -1)

We built models based on the above hyper-parameter space, evaluated each model using 5-fold cross validation and selected the parameter combination which produced the best results. The scoring metric used was accuracy.

d) Best Parameters Learned Using Grid Search: The best parameters learned using Grid Search are:

{'min_samples_split': 0.2, 'splitter': 'best', 'criterion': 'gini', 'max_depth': 2, 'min_samples_leaf': 0.04}

e) Accuracy of Model made with Best Parameters learned:

1) Accuracy of Model with Best Parameters (Hyper Parameter Tuning) using 5-fold cross validation:

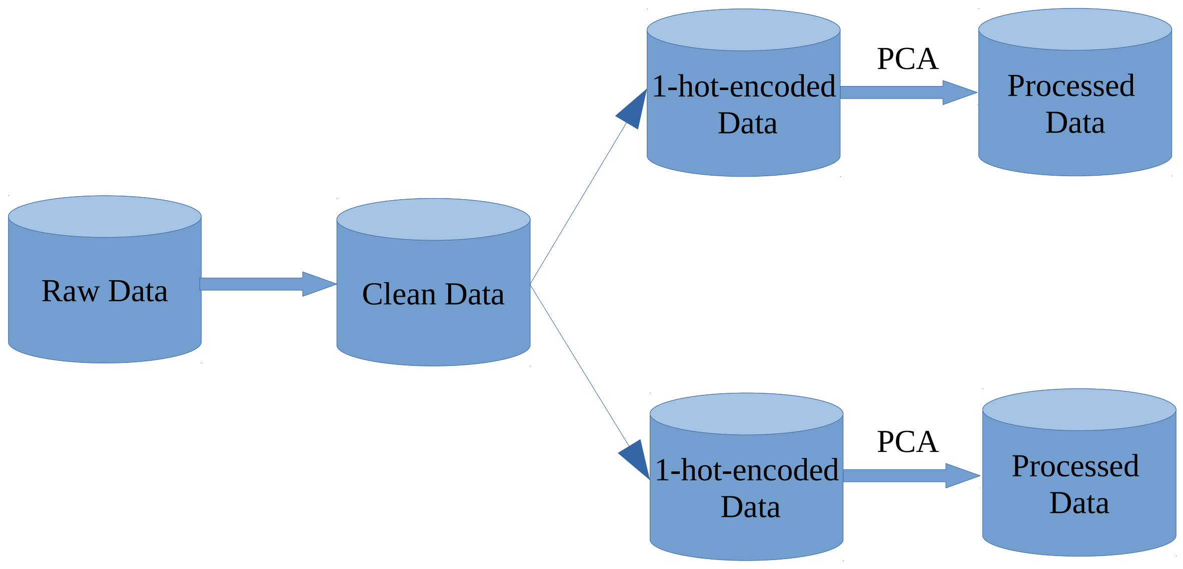
Accuracy = 55.1 %

2) Accuracy of Model with Best Parameters (Hyper Parameter Tuning) on the Test Set:

Accuracy = 59.2 %

SVM(Support Vector Machines)

Data Preprocessing



Drawing 1: Data Processing procedure

While running SVM on our dataset, the first question comes into mind whether we should do SVM on categorical data. The answer is we should convert categorical data into numerical before running SVM [6]. As the feature values do not contains orders among them, we convert them first into 1-hot-encoding. So the dimension of the data is fueled up. To reduce the dimension we run PCA on it. *Drawing 1* shows the procedure of data processing for SVM.

Table 4 shows how dimension of the data is first fueled up when we do 1-hot-encoding and then reduced down when we run PCA. Note that, we did two 1-hot-encoding to see how much the dimension is going to change. If we take only categorical attributes from cleaned dataset instead of taking all attributes the dimension can be reduced almost 11%. The intuition behind this is only diagnosis result should have more impact on class attribute and that is categorical. In *Table 4*, "Only categorical column" consists of all diagnosis result attributes, race, gender and age range.

Raw	Cleaned	1-hot-encoded (taking all columns)	Keeping variance (σ *)*	After PCA	Only categorical attributes	1-hot-encoded (taking only categorical columns)	Keeping variance (σ *)*	After PCA
101766, 50	101766, 40	101766, 2670	.80	101766, 1888	101766, 33	101766, 2354	.80	101766, 1666
			.90	101766, 2204			.90	101766, 1941
			.95	101766, 2381			.95	101766, 2098

Table 4: 1-hot-encoding and PCA on attributes for dimensionality reduction.

Subsampling technique

The dominating cost of running SVM is $O\left(n_{features} X\, n_{observations}^2\right)$ where in our case *features>=1666 and observations=101766*. So it is obvious that the running cost is very high so we follow subsampling technique in which for every dataset after running PCA we take .1%, 1% and 10% of the data and divide them into 20% test data and 80% train data, and find out if there is any significant increase of accuracy with the increase of data by 10 multitutde. So far our hyperparameter is *keep_variance* denoted by σ while running PCA and *subsample_size denoted by n*.

Table 5, 7 and 9 show mean accuracy, precision, recall and f1-score on the dataset that was produced by running PCA on all attributes while σ was 80%, 90% and 95% respectively. On the other hand *Table 6, 8 and 10* shows 5-fold cross validation score on the dataset that was produced using same process.

σ	n	Data distribution	Support	Accuracy	Precision	Recall	f1-score
80%	.1%	Train: 81, 1888	NO: 43 >30: 26 <30: 12	0.57	0.85	0.39	0.34
		Test: 21, 1888	NO: 7 >30: 12 <30: 2	0.33	0.11	0.33	0.17
	1%	Train: 814, 1888	NO: 450 >30: 264 <30: 100	0.61	0.86	0.41	0.39
		Test: 204, 1888	NO: 124 >30: 54 <30: 26	0.61	0.20	0.33	0.25
	10%	Train: 8141, 1888	NO: 4434 >30: 2857 <30: 850	0.61	0.86	0.43	0.42
		Test: 2036, 1888	NO: 1098 >30: 731 <30: 207	0.53	0.32	0.33	0.25

Table 5: Accuracy, Precision, Recall and f1-score where σ=80% (all attributes)

σ	n	Data distribution	1-fold	2-fold	3-fold	4-fold	5-fold
80%	.1%	102, 1888	0.333	0.333	0.333	0.333	0.333
	1%	1018, 1888	0.330	0.330	0.333	0.333	0.330
	10%	10177, 1888	0.332	0.343	0.335	0.337	0.334

Table 6: 5-fold cross validation where σ=80% (all attributes)

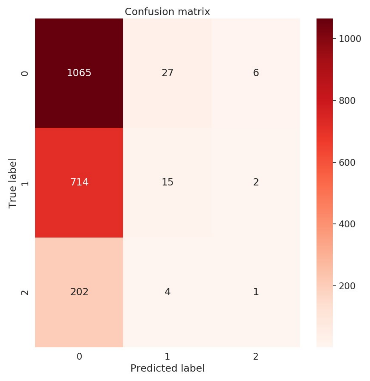


Figure 6: σ=80%, n=10%

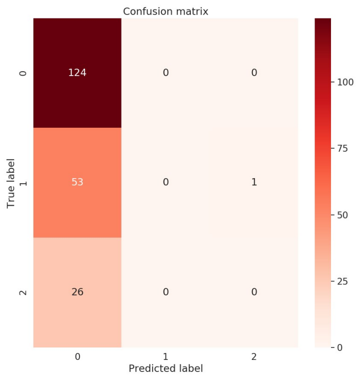


Figure 7: σ=80%, n=1%

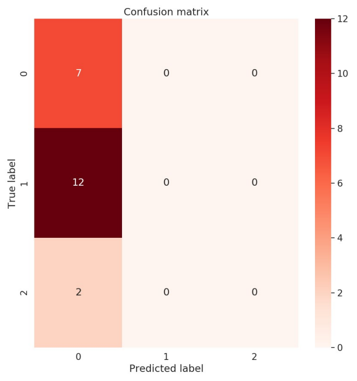


Figure 8: σ=80%, n=.1%

σ	n	Data distribution	Support	Accuracy	Precision	Recall	f1-score
90%	.1%	Train: 81, 2204	NO: 43 >30: 32 <30: 6	0.58	0.85	0.47	0.44
		Test: 21, 2204	NO: 9 >30: 9 <30: 3	0.42	.14	.33	.20
	1%	Train: 814, 2204	NO: 420 >30: 298 <30: 96	.60	.85	.45	.44
		Test: 204, 2204	NO: 109 >30: 70 <30: 25	.53	.18	.33	.23
	10%	Train: 8141, 2204	NO: 4383 >30: 2858 <30: 900	.61	.86	.44	.43
		Test: 2036, 2204	NO: 1075 >30: 745 <30: 216	.52	.33	.33	.25

Table 7: Accuracy, Precision, Recall and f1-score where σ=90% (all attributes)

σ	n	Data distribution	1-fold	2-fold	3-fold	4-fold	5-fold
90%	.1%	102, 2204	0.333	0.333	0.333	0.333	0.333
	1%	1018, 2204	0.333	0.324	0.338	0.330	0.333
	10%	10177, 2204	0.339	0.330	0.337	0.330	0.331

Table 8: 5-fold cross validation where σ=90% (all attributes)

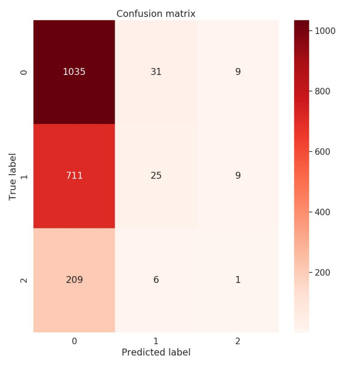


Figure 9: σ=90%, n=10%

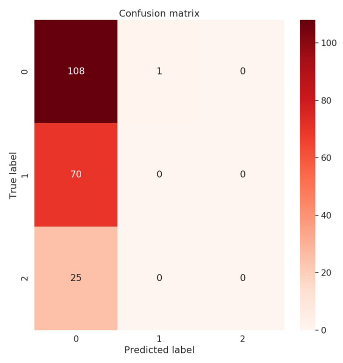


Figure 10: σ=90%, n=1%

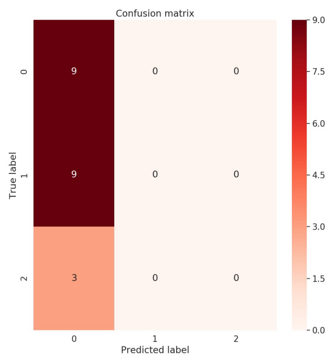


Figure 11: σ=90%, n=.1%

σ	n	Data distribution	Support	Accuracy	Precision	Recall	f1-score
95%	.1%	Train: 81, 2381	NO: 53 >30: 25 <30: 3	.69	.56	.37	.34
		Test: 21, 2381	NO: 12 >30: 6 <30: 3	.57	.19	.33	.24
	1%	Train: 814, 2381	NO: 444 >30: 289 <30: 81	.62	.86	.44	.43
		Test: 204, 2381	NO: 120 >30: 61 <30: 23	.59	.20	.33	.25
	10%	Train: 8141, 2381	NO: 4427 >30: 2859 <30: 855	.61	.85	.43	.41
		Test: 2036, 2381	NO: 1095 >30: 719 <30: 222	.53	.31	.33	.25

Table 9: Accuracy, Precision, Recall and f1-score where σ=95% (all attributes)

σ	n	Data distribution	1-fold	2-fold	3-fold	4-fold	5-fold
95%	.1%	102, 2381	0.333	0.333	0.333	0.333	0.333
	1%	1018, 2381	0.330	0.333	0.330	0.342	0.330
	10%	10177, 2381	0.330	0.335	0.332	0.335	0.336

Table 10: 5-fold cross validation where σ=95% (all attributes)

Table 11 and 13 shows mean accuracy, precision, recall and f1-score on the dataset that was produced by running PCA on one-hot encoding of only diagnosis attributes while the minimum variance to keep was 80% and 90%. On the other hand Table 12 and 14 shows 5-fold cross validation score on the dataset that was produced using same process.

σ	n	Data distribution	Support	Accuracy	Precision	Recall	f1-score
80%	.1%	Train: 81, 1666	NO: 40 >30: 30 <30: 11	.56	.84	.45	.41
		Test: 21, 1666	NO: 14 >30: 4 <30: 3	.67	.22	.33	.27
	1%	Train: 814, 1666	NO: 420 >30: 298 <30: 96	.58	.85	.42	.39
		Test: 204, 1666	NO: 124 >30: 62 <30: 18	.60	.20	.33	.25
	10%	Train: 8141, 1666	NO: 4354 >30: 2838 <30: 949	.59	.84	.42	.39
		Test: 2036, 1666	NO: 1098 >30: 714 <30: 224	.53	.33	.333	.25

Table 11: Accuracy, Precision, Recall and f1-score on 80% explained variance PCA (diagnosis attributes)

σ	n	Data distribution	1-fold	2-fold	3-fold	4-fold	5-fold
80%	.1%	102, 1666	0.333	0.333	0.333	0.333	0.333
	1%	1018, 1666	0.337	0.324	0.333	0.330	0.327
	10%	10177, 1666	0.332	0.337	0.327	0.336	0.328

Table 12: 5-fold cross validation on 80% explained variance PCA (diagnosis attributes)

σ	n	Data distribution	Support	Accuracy	Precision	Recall	f1-score
90%	.1%	Train: 81, 1942	NO: 49 >30: 25 <30: 7	.68	.88	.52	.53
		Test: 21, 1942	NO: 9 >30: 9 <30: 3	.43	.14	.33	.20
	1%	Train: 814 , 1942	NO: >30: <30:	.58	.85	.43	.40
		Test: 204, 1942	NO: >30: <30:	.54	.18	.33	.23
	10%	Train: 8141, 1942	NO: >30: <30:	.61	.85	.43	.42
		Test: 2036 , 1942	NO: >30: <30:	.52	.31	.33	.25

Table 13: Accuracy, Precision, Recall and f1-score on 90% explained variance PCA (diagnosis attributes)

σ	n	Data distribution	1-fold	2-fold	3-fold	4-fold	5-fold
90%	.1%	102, 1942	0.333	0.333	0.333	0.333	0.333
	1%	1018, 1942	0.337	0.327	0.330	0.333	0.333
	10%	10177, 1942	0.332	0.327	0.339	0.334	0.333

Table 14: 5-fold cross validation on 90% explained variance PCA (diagnosis attributes)

From the general, we decide to keep two dataset for further SVM parameter tuning that we call *95-pca-all* and 90-pca-diag. Note that, *95-pca-all* means the dataset which was produced by taking all attributes and keeping variance 95% while running PCA for dimensionality reduction. And *90-pca-diag* means the dataset which was produced by taking only diagnosis result attributes and keeping variance 90% while running PCS for dimensionality reduction.

References:

1. https://archive.ics.uci.edu/ml/index.php
2. https://www.analyticsvidhya.com/blog/2018/05/24-ultimate-data-science-projects-to-boost-your-knowledge-and-skills/
3. https://www.dataoptimal.com/data-science-projects-2018/
4. https://archive.ics.uci.edu/ml/datasets/diabetes+130-us+hospitals+for+years+1999-2008
5. https://www.hindawi.com/journals/bmri/2014/781670/
6. https://www.quora.com/How-can-I-apply-an-SVM-for-categorical-data