ANLY 530 Machine Learning I, Laboratory #2

Introduction

As you learned in ANLY 500 analytics can be categorized as descriptive, predictive or prescriptive. Machine learning is engaged in all categories of analytics. We've obviously done descriptive and predictive analytics. Prescriptive analytics is what happens when you take predictions made and use them to make strategic changes, e.g. to a business model in order to refocus or enhance the model. Part of this process is also handling risk. Analytics is frequently used to consider a variety of types of risk; credit, sales, fraud related, etc. One of the challenges to prescriptive analytics is the complexity in integrating multiple streams of data, e.g. business, risk, environmental, contextual, stakeholder, and so much more.

This laboratory will get you started thinking about pieces of what is required for prescriptive analytics and in particular how machine learning can be used to support that function. This laboratory will specifically look at credit risk. We'll define credit risk as the risk of default on a debt due to a borrower failing to make the required payments in a timely manner. In this definition, the lender assumes the risk of losing both the principal and the anticipated interest on that principal. For the problems in this laboratory we will use a bank as the financial institution. The bank will analyze customer data to predict which customers might be credit risks. These predictions will then feed into risk management.

This particular laboratory focuses on the use of Bayesian methods, specifically Naïve Bayesian Classifiers. You won't necessarily have to read the uploaded papers or the websites listed below to complete this laboratory. They are available only if you want more information. These two papers have been uploaded to Moodle and are intended to provide you with additional background and information about the different challenges and algorithms used for credit scoring; "Guide to Credit Scoring in R' by Sharma, and "Classification methods applied to credit scoring: A systematic review and overall comparison" by Louzada. You may use following links to learn more about Bayesian network learning and its usage in scikit-learn package:

- https://scikit-learn.org/stable/modules/naive_bayes.html
- https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/

A data set has been provided for you, creditData.csv. We've used many data sets already. You'll need to **start by loading the data into Python and looking at the data set**. Since we've done this many times I won't include a separate step for reading in the data and examining it in this laboratory. That is, the equivalent of "Step 1" in Laboratory #1 and is not explicitly included in this laboratory. A description of the data set is included at the end of this laboratory.

One of the differences between data mining and machine learning is that machine learning focuses more on improving performance, whether that is improving the performance of an algorithm or increasing learning, than it (machine learning) does on the specific results obtained from running an algorithm (as does data mining). In this laboratory we'll begin to consider that and take a "brute force" approach to improving performance and/or increasing learning. We'll begin by using Python functions for a Naïve Bayes Classifier as black-box tools and then we'll apply common techniques for improvement. I'll call the original, unimproved application of the algorithms Part 1. The brute force part is that we'll apply techniques to improve performance individually and manually to the original algorithms. I'll call this Part 2.

Laboratory 2: Naïve Bayes Classifiers, Part 1

Step 1: Exploring and Preparing the Data For many functions to work correctly there cannot be any missing values in the data set. So far we haven't used any data sets that require a lot of processing or looking to determine if there are missing values. For this data set it is more difficult to just visually ensure that there are no missing values. So, you could use the following command built on the *sum()* and *isnull()* functions (in newer versions of python, you can use *isna()* command). It will return a value that is the sum (total number of) all missing values:

```
credit.isnull().sum()
```

Q1- What is your suggestion if you see any NA values?

Next, use a 75%/25% split for training and test data, i.e. use 75% of the records for the training set and 25% of the records for the test set. Report the number of missing values you find in the data in your results report. Use the randomization seed of 123.

Q2- Compute the percentage of both classes similar to what you did in lab 1 and see if the distribution of both classes preserved for both training and testing data.

Step 2: Training a Model on the Data Use the `GaussianNB()` function on your training data to build the Naïve Bayes Classifier.

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb = gnb.fit(X_train, y_train)
```

Step 3: Evaluating Model Performance Now, use the processes we have completed before to evaluate your Naïve Bayes Classifier. (Hint: If you didn't convert your class variable before you will not be able to complete this step!) In your results report include your confusion table

showing your results. Be sure to include 2-3 sentences explaining what those results actually mean.

Laboratory 2: Naïve Bayes Classifiers, Part 2

In this part of the laboratory we'll manually work to improve the performance of the Naïve Bayes classifier. There are several ways of improving the performance of an algorithm, e.g. pre-processing the data can make a very big difference in the performance of any algorithm. One category of methods to improve performance of algorithms is referred to as feature selection. We will follow the approach using correlations to determine if there are variables in the data set that if they are eliminated would improve the performance of the Naïve Bayes Classifier. You can choose whatever method you want to use to select the features (variables) to keep. Be sure to include a paragraph or so in your results report describing what/how you tried to increase the Naïve Bayes algorithm's performance; and, include any gains in performance in that report too.

Step 1: Exploring and Preparing the Data The first approach I'll introduce is using the correlation matrix to perform feature, i.e. variable selection. As always, there are several packages with functions for computing a correlation matrix. But, before doing anything you need to determine if any pre-processing of the data is required, e.g. scaling. Be sure to report any pre-processing of the data in your results report

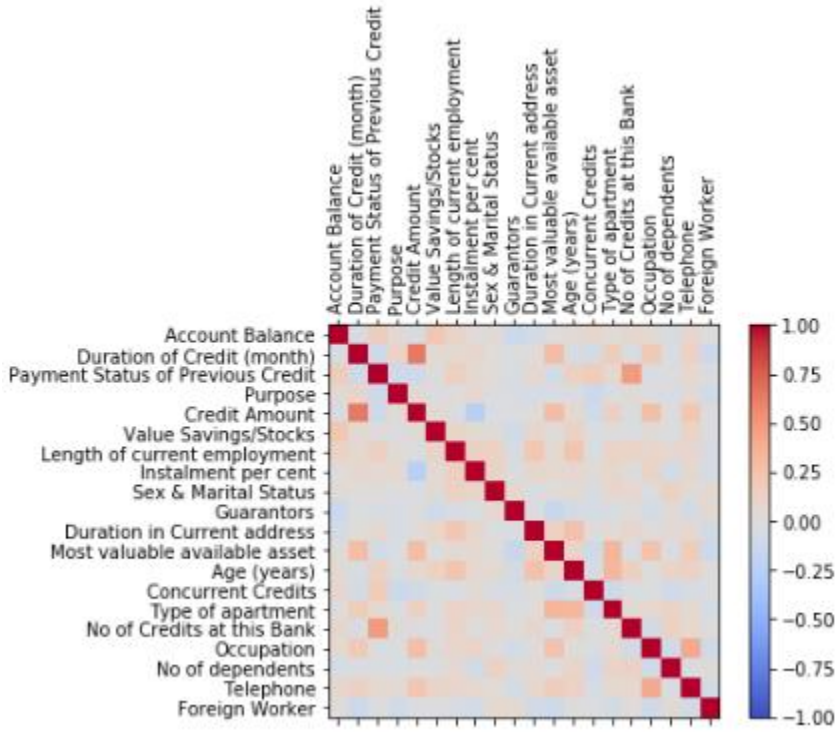
Next, you'll need to pull out those features, or variables, with correlations higher than a specified threshold. What is the best threshold to use? It depends on whether you're most interested in achieving the highest level of classification with the fewest errors (increasing performance) or in balancing this with the variables that reduce overall risk the most. That is, if you eliminate too many features (variables) you may end up with a classifier that misses some critical piece of evidence that actually increases risk even though the classifier performs more perfectly.

Let's start with using following code to see how correlated the variables are:

```
import matplotlib.pyplot as plt

corr = X_train.corr()
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(corr,cmap='coolwarm', vmin=-1, vmax=1)
fig.colorbar(cax)
ticks = np.arange(0,len(X_train.columns),1)
ax.set_xticks(ticks)
plt.xticks(rotation=90)
ax.set_yticks(ticks)
ax.set_xticklabels(X_train.columns)
ax.set_yticklabels(X_train.columns)
plt.show()
```

By running this code, you will see this graph.



You can print the variable *corr* to see actual value of correlations.

Now you can use following code to remove the features that have correlation over 0.6.

```
# Create correlation matrix
corr_matrix = X.corr().abs()

# Select upper triangle of correlation matrix
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape),
k=1).astype(np.bool))

# Find index of feature columns with correlation greater than 0.60
to_drop = [column for column in upper.columns if any(upper[column] > 0.6)]

Xnew = X.drop(to_drop, axis=1)
```

Split your data in the similar way that you did in previous section.

Step 2: Training a Model on the Data At this point we follow the same process that we have before. To train the model

Step 3: Evaluating Model Performance

Evaluate the Naïve Bayes Classifier as usual

Q3- What is the accuracy this time? Be sure to include in your results report whether or not, after all this work, the performance of your Naïve Bayes Classifier was improved.

Laboratory 2: Support Vector Machine, Part 3

Step 1: Collecting the Data

We'll look at support vector machines from the perspective of optical character recognition (OCR).We'll use the letterdata.csv file also from the UCI Data Repository.We can simply read it in and look at the structure as follows:

```
input_file = "address/of/the/file/letterdata.csv"
letters = pd.read_csv(input_file)
letters.head(5)
```

	letter	xbox	ybox	width	height	onpix	xbar	ybar	x2bar	y2bar	xybar	x2ybar	xy2bar	xedge	xedgey	yedge	yedgex
0	T	2	8	3	5	1	8	13	0	6	6	10	8	0	8	0	8
1	I	5	12	3	7	2	10	5	5	4	13	3	9	2	8	4	10
2	D	4	11	6	8	6	10	6	2	6	10	3	7	3	7	3	9
3	N	7	11	6	6	3	5	9	4	6	4	4	10	6	10	2	8
4	G	2	1	3	1	1	8	6	6	6	6	5	9	1	7	5	10

Step 2: Preparing the Data

The only thing we need to do to prepare this data set is to subset it into our training and test data sets. There are 20,000 observations so we have considerable flexibility in deciding how many to put in our training set and how many to keep for testing. Let's use 90% of this data for training and the rest for testing. Use any of the methods that you have learnt by now.

Step 3: Training a Model on the Data

Use the same methodology that you saw in class and design a linear SVM to make classification.

```
from sklearn import svm
# Design the model
clf = svm.SVC(kernel='linear')
#clf = svm.LinearSVC(C=1)
clf.fit(X_train, y_train)
y_predict = clf.predict(X_test)
```

Step 4: Evaluating Model Performance

Again, we'll develop a confusion table to determine how well our model performed overall. Since we are looking at letters of the alphabet we'll have a considerable number of columns and rows to our table. You might want to adjust the size of the console frame accordingly. The commands to run the evaluation and output the table is:

```
confusion_matrix(y_test, y_predict)
```

```
array([[71,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0, 0,1,0,0,0,0,0,0,2,0],
       [1,67,0,1,0,0,2,1,0,0,1,0,0,0,0,0,0,2,1,0,0,0,0,0,0,0],
       [0,0,67,0,2,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0],
       [0,5,0,67,0,0,0,1,0,1,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0],
       [0,1,1,0,58,1,3,0,1,0,1,4,0,0,0,0,2,0,1,3,0,0,0,0,0,2],
       [0,1,1,0,0,74,0,0,0,1,0,0,0,2,0,1,0,0,3,2,0,0,0,0,1,0],
       [0,0,4,3,1,0,61,0,0,0,1,2,0,0,0,1,2,0,4,0,0,0,1,0,0,0],
       [0,1,0,0,0,2,0,52,0,1,1,1,0,0,7,0,0,3,0,0,2,0,0,1,0,0],
       [0,1,0,1,0,2,0,0,67,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1],
       [2,0,0,0,0,1,0,0,5,67,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
       [0,1,2,1,0,0,3,2,0,0,68,0,0,0,0,0,0,3,0,1,0,0,0,2,0,0],
       [0,0,0,0,1,0,1,1,0,0,1,63,0,0,0,0,2,0,1,0,0,0,0,0,0,0],
       [0,1,0,0,0,0,0,3,0,0,0,0,83,0,1,0,0,2,0,0,1,0,0,0,0,0],
       [0,0,0,0,0,0,0,1,0,0,1,0,0,65,1,0,0,1,0,0,0,1,0,0,0,0],
       [1,0,1,1,0,0,0,7,0,0,0,0,1,0,66,0,1,1,0,0,1,0,1,0,0,0],
       [0,0,0,0,0,9,0,0,0,0,0,0,0,0,0,69,0,0,0,0,0,0,1,0,2,0],
       [3,0,0,1,0,0,1,2,0,0,1,0,0,0,3,1,49,0,4,0,0,1,0,0,0,0],
       [1,3,1,1,0,0,1,1,0,0,1,0,0,0,1,0,0,62,0,0,0,0,0,0,0,0],
       [0,3,0,0,2,1,2,0,1,0,0,1,0,0,0,0,4,0,51,0,0,0,0,1,0,6],
       [0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,1,76,0,0,0,0,3,0],
       [0,0,0,0,0,0,0,1,0,0,0,0,1,0,1,0,0,0,0,0,70,0,0,0,0,0],
       [1,1,0,0,0,0,1,1,0,0,0,0,1,0,0,0,0,0,0,0,0,69,0,0,4,0],
       [0,0,0,0,0,0,1,0,0,0,0,0,4,1,1,0,0,0,0,0,3,1,68,0,0,0],
       [0,0,0,0,2,0,0,0,1,1,2,2,0,0,0,0,0,0,0,0,0,0,0,73,1,0],
       [1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,3,76,0],
       [0,0,0,0,2,0,0,0,0,3,0,0,0,0,0,0,0,0,7,0,0,0,0,0,0,62]], dtype=
int64)
```

Now compute the accuracy.

Q4- We may be able to do better than this by changing the Kernels. Try Polynomial and RBF kernels to improve the result.

Laboratory 2: News popularity, Part 4

Now apply the Naïve Bayes classifier and SVM that you saw in Parts 1 through 3 on News popularity data set from lab 1.

Running SVM on original dataset will take a long time. To resolve this issue, normalize the data and/or try to remove some of the features or consider less number of instances for training.

Q5- Do you see any improvement compared to last three techniques? Please completely explain your results and analysis.

Summary of covariates for the dataset "Determining the solidness of borrowers via creditscoring"

Notice:
The given score for the categorical covariates is based on the assessment of experienced bank specialists dealing with credits and is contained in the downloadable file. The description of the variables is consistent with "Fahrmeir / Hamerle / Tutz (1996, 2nd ed.): *Multivariate statistische Verfahren.* de Gruyter, Berlin. p. 390 ff."

Variable	Description	Categories	Score	rel. frequency in % for	
				good credits	bad credits
Creditability	Creditability: 1 : credit-worthy 0 : not credit-worthy				
Account Balance	Balance of current account	no balance or debit	2	35.00	23.43
		0 <= ... < 200 DM	3	4.67	7.00
		... >= 200 DM or checking account for at least 1 year	4	15.33	49.71
		no running account	1	45.00	19.86
Duration of Credit (month)	Duration in months (metric)				
Duration of Credit (month)	Duration in months (categorized)	<=6	10	3.00	10.43
		6 < ... <= 12	9	22.33	30.00
		12 < ... <= 18	8	18.67	18.71
		18 < ... <= 24	7	22.00	22.57
		24 < ... <= 30	6	6.33	5.43
		30 < ... <= 36	5	12.67	6.86
		36 < ... <= 42	4	1.67	1.71
		42 < ... <= 48	3	10.67	3.14
		48 < ... <= 54	2	0.33	0.14
		> 54	1	2.33	1.00
Payment Status of Previous	Payment of previous credits	no previous credits / paid back all previous	2	56.33	51.57

Credit		credits			
		paid back previous credits at this bank	4	16.67	34.71
		no problems with current credits at this bank	3	9.33	8.57
		hesitant payment of previous credits	0	8.33	2.14
		problematic running account / there are further credits running but at other banks	1	9.33	3.00
Purpose	Purpose of credit	new car	1	5.67	12.29
		used car	2	19.33	17.57
		items of furniture	3	20.67	31.14
		radio / television	4	1.33	1.14
		household appliances	5	2.67	2.00
		repair	6	7.33	4.00
		education	7	0.00	0.00
		vacation	8	0.33	1.14
		retraining	9	11.33	9.00
		business	10	1.67	1.00
		other	0	29.67	20.71
Credit Amount	Amount of credit in "Deutsche Mark" (metric)				
Credit Amount	Amount of credit in DM (categorized)	<=500	10	1.00	2.14
		500 < ... <= 1000	9	11.33	9.14
		1000 < ... <= 1500	8	17.00	19.86
		1500 < ... <= 2500	7	19.67	24.57
		2500 < ... <= 5000	6	25.00	28.57
		5000 < ... <= 7500	5	11.33	9.71
		7500 < ... <= 10000	4	6.67	3.71
		10000 < ... <= 15000	3	7.00	2.00
		15000 < ... <= 20000	2	1.00	0.29
		> 20000	1	0.00	0.00

Value Savings/Stocks	Value of savings or stocks	< 100,- DM	2	11.33	9.86
		100,- <= ... < 500,- DM	3	3.67	7.43
		500,- <= ... < 1000,- DM	4	2.00	6.00
		>= 1000,- DM	5	10.67	21.57
		not available / no savings	1	72.33	55.14
Length of current employment	Has been employed by current employer for	unemployed	1	7.67	5.57
		<= 1 year	2	23.33	14.57
		1 <= ... < 4 years	3	34.67	33.57
		4 <= ... < 7 years	4	13.00	19.29
		>= 7 years	5	21.33	27.00
Instalment per cent	Installment in % of available income	>= 35	1	11.33	14.57
		25 <= ... < 35	2	20.67	24.14
		20 <= ... < 25	3	15.00	16.00
		< 20	4	53.00	45.29
Sex & Marital Status	Marital Status / Sex	male: divorced / living apart	1	6.67	4.29
		female: divorced / living apart / married	2	11.33	10.29
		male: single	2	25.00	18.43
		male: married / widowed	3	48.67	57.43
		female: single	4	8.33	9.57
Guarantors	Further debtors / Guarantors	none	1	90.67	90.71
		Co-Applicant	2	6.00	3.29
		Guarantor	3	3.33	6.00
Duration in Current address	Living in current household for	< 1 year	1	12.00	13.43
		1 <= ... < 4 years	2	32.33	30.14
		4 <= ... < 7 years	3	14.33	15.14
		>= 7 years	4	41.33	41.29
Most valuable available asset	Most valuable available assets	Ownership of house or land	4	22.33	12.43

		Savings contract with a building society / Life insurance	3	34.00	32.86
		Car / Other	2	23.67	23.00
		not available / no assets	1	20.00	31.71
Age (years)	Age in years (metric)				
Age (years)	Age in years (categorized)	0 <= ... <= 25	1	26.67	15.71
		26 <= ... <= 39	2	47.33	52.72
		40 <= ... <= 59	3	21.67	26.14
		60 <= ... <= 64	5	2.33	3.00
		>= 65	4	2.00	2.43
Concurrent Credits	Further running credits	at other banks	1	19.00	11.71
		at department store or mail order house	2	6.33	4.00
		no further running credits	3	74.67	84.29
Type of apartment	Type of apartment	rented flat	2	62.00	75.43
		owner-occupied flat	3	14.67	9.14
		free apartment	1	23.33	15.57
No of Credits at this Bank	Number of previous credits at this bank (including the running one)	one	1	66.67	61.86
		two or three	2	30.67	34.43
		four or five	3	2.00	3.14
		six or more	4	0.67	0.57
Occupation	Occupation	unemployed / unskilled with no permanent residence	1	2.33	2.14
		unskilled with permanent residence	2	18.67	20.57
		skilled worker / skilled employee / minor civil servant	3	62.00	63.43
		executive / self-employed / higher civil servant	4	17.00	13.86
No of dependents	Number of persons	0 to 2	2	84.67	84.43

	entitled to maintenance	3 and more	1	15.33	15.57
Telephone	Telephone	no	1	62.33	58.43
		yes	2	37.67	41.57
Foreign Worker	Foreign worker	yes	1	1.33	4.71
		no	2	98.67	95.29

The covariates M1, M2, M5, M7, M8 are consistent with the description in "Fahrmeir / Hamerle / Tutz (1985, 1st ed.): *Multivariate statistische Verfahren.* de Gruyter, Berlin. p. 285 ff."