Artificial Intelligence
Data analysis and comparison using 2 data sets and 4 classifier models

2013150133 Jeongho Yoon

Abstract

This experiment aims to make various classifier models using two data sets, Bank marketing data and Dota2 play data, analyze the data itself, and compare the models.

The development of machine learning is causing huge changes in the way people live. Recommendations in Internet shopping or web contents consumption is quite common, AI lawyers are being recruited by law firms, and autonomous vehicles are coming closer to being an everyday part of life.

Machine learning is opposite in direction with past artificial intelligence technology, such as expert systems, which makes use of deductive reasoning. It lets machines make use of, or at least simulate, inductive reasoning, which was deemed to be the domain of living organisms.

While deductive reasoning is the inference of instances with reference to a more general rule, inductive reasoning is the inference of the general rule with reference to instances. Therefore, for inductive reasoning to be effective, there needs to be instances, whether the trainee is an animal or a machine. From here are raised two important questions regarding data. Firstly, how do we collect quantitatively larger and qualitatively better data? Secondly, how do we better make use of collected data?

This experiment was designed to address the second question. There can be many ways in which data can be used optimally. Using algorithms that match the characteristics of the data collected can be one. Other ways include changing various values in the algorithms – such as changing the number of hidden layers in multi layered perceptrons – or data preprocessing, such as feature selection.

Another point of this experiment is to determine if machine learning algorithms could be used to build a better understanding of the data set itself. For example, we may be interested in knowing which features of an animal is most important in deciding its family, and this understanding may be better reached through models built by machine learning algorithms. Although some models, such as multi layered perceptrons, are known to be hard to figure out what the model actually means, other models, such as regression models or decision tree models, can help us out greatly.

With this in mind, two data sets and four classifier models were selected. The reason two data sets instead of one were used is because there are vast differences among data sets. For example, some data sets have primarily numeric attributes, and others primarily have categorical attributes. By choosing multiple data sets, it could be confirmed if a method that worked in one data set worked in other data sets as well or not. Also, the reason multiple classifier models were used was in order to find the optimal model, compare different models, and also because different models can provide different analysis about data.

The data sets used were Bank Marketing data and Dota2 data, both of which are from UCI machine learning data repository. The reason Bank Marketing data was chosen was because it had an adequate number of attributes, abundant instances, and contained many types of attributes, which are rare qualities of free data sets.

The reason Dota2 data was used is because it also has a good number of attributes and instances, with many types of attributes. This data set is also interesting because a game is an artificially made environment where interactions between many variables are either magnified or unmagnified.

The bank marketing data set is a data set that recorded various attributes of clients, and whether they subscribed to a product that was directly marketed over the phone. A total of 45211 instances were recorded, and 16 attributes of each client was collected. These attributes include

1 - age (numeric)
2 - job : type of job (categorical: "admin.","unknown","unemployed","management","housemaid","entrepreneur","student", "blue-collar","self-employed","retired","technician","services")
3 - marital : marital status (categorical: "married","divorced","single"; note: "divorced" means divorced or widowed)
4 - education (categorical: "unknown","secondary","primary","tertiary")
5 - default: has credit in default? (binary: "yes","no")
6 - balance: average yearly balance, in euros (numeric)
7 - housing: has housing loan? (binary: "yes","no")
8 - loan: has personal loan? (binary: "yes","no")
related with the last contact of the current campaign:
9 - contact: contact communication type (categorical: "unknown","telephone","cellular")
10 - day: last contact day of the month (numeric)
11 - month: last contact month of year (categorical: "jan", "feb", "mar", ..., "nov", "dec")
12 - duration: last contact duration, in seconds (numeric)
other attributes:
13 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
14 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric, -1 means client was not previously contacted)
15 - previous: number of contacts performed before this campaign and for this client (numeric)
16 - poutcome: outcome of the previous marketing campaign (categorical: "unknown","other","failure","success")

And the final column recorded whether this client subscribed to the product or not.

The library used in this experiment is SciKitLearn. It is a python machine learning library which, in m any ways, is simpler to use than Tensorflow.

The classifier models used in this experiment are logistic regression, decision tree, naive bayesian, and multi layered perceptron, all of which are provided in SciKitLearn.

Data exploration

The CSV file was read in using Pandas, which is a python library for data analysis. And then the data was printed out to see the overview.

```
bankData=pd.read_csv("directory", sep=';')
print bankData
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	79	1	-1	0	unknown	no
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may	220	1	339	4	failure	no
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	185	1	330	1	failure	no
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	4	-1	0	unknown	no
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	226	1	-1	0	unknown	no
5	35	management	single	tertiary	no	747	no	no	cellular	23	feb	141	2	176	3	failure	no
6	36	self-employed	married	tertiary	no	307	yes	no	cellular	14	may	341	1	330	2	other	no
7	39	technician	married	secondary	no	147	yes	no	cellular	6	may	151	2	-1	0	unknown	no
8	41	entrepreneur	married	tertiary	no	221	yes	no	unknown	14	may	57	2	-1	0	unknown	no
9	43	services	married	primary	no	-88	yes	yes	cellular	17	apr	313	1	147	2	failure	no
10	39	services	married	secondary	no	9374	yes	no	unknown	20	may	273	1	-1	0	unknown	no
11	43	admin.	married	secondary	no	264	yes	no	cellular	17	apr	113	2	-1	0	unknown	no
12	36	technician	married	tertiary	no	1109	no	no	cellular	13	aug	328	2	-1	0	unknown	no
13	20	student	single	secondary	no	502	no	no	cellular	30	apr	261	1	-1	0	unknown	yes
14	31	blue-collar	married	secondary	no	360	yes	yes	cellular	29	jan	89	1	241	1	failure	no
15	40	management	married	tertiary	no	194	no	yes	cellular	29	aug	189	2	-1	0	unknown	no
16	56	technician	married	secondary	no	4073	no	no	cellular	27	aug	239	5	-1	0	unknown	no
17	37	admin.	single	tertiary	no	2317	yes	no	cellular	20	apr	114	1	152	2	failure	no
18	25	blue-collar	single	primary	no	-221	yes	no	unknown	23	may	250	1	-1	0	unknown	no
19	31	services	married	secondary	no	132	no	no	cellular	7	jul	148	1	152	1	other	no
20	38	management	divorced	unknown	no	0	yes	no	cellular	18	nov	96	2	-1	0	unknown	no
21	42	management	divorced	tertiary	no	16	no	no	cellular	19	nov	140	3	-1	0	unknown	no
22	44	services	single	secondary	no	106	no	no	unknown	12	jun	109	2	-1	0	unknown	no
23	44	entrepreneur	married	secondary	no	93	no	no	cellular	7	jul	125	2	-1	0	unknown	no
24	26	housemaid	married	tertiary	no	543	no	no	cellular	30	jan	169	3	-1	0	unknown	no
25	41	management	married	tertiary	no	5883	no	no	cellular	20	nov	182	2	-1	0	unknown	no
26	55	blue-collar	married	primary	no	627	yes	no	unknown	5	may	247	1	-1	0	unknown	no
27	67	retired	married	unknown	no	696	no	no	telephone	17	aug	119	1	105	2	failure	no
28	56	self-employed	married	secondary	no	784	no	yes	cellular	30	jul	149	2	-1	0	unknown	no
29	53	admin.	married	secondary	no	105	no	yes	cellular	21	aug	74	2	-1	0	unknown	no
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4491	35	blue-collar	single	secondary	no	0	yes	no	cellular	16	apr	169	1	-1	0	unknown	no
4492	32	technician	single	secondary	no	309	yes	yes	cellular	16	apr	346	1	234	3	failure	no
4493	28	technician	single	tertiary	no	0	yes	no	unknown	4	jun	205	6	-1	0	unknown	no
4494	26	technician	single	secondary	no	668	yes	no	unknown	28	may	576	3	-1	0	unknown	yes
4495	48	management	married	tertiary	no	1175	yes	no	telephone	18	nov	1476	3	-1	0	unknown	no
4496	30	blue-collar	single	secondary	no	363	no	no	cellular	28	jul	171	3	-1	0	unknown	no
4497	31	entrepreneur	single	tertiary	no	38	no	no	cellular	20	nov	185	2	-1	0	unknown	no
4498	31	management	married	tertiary	no	1183	yes	no	unknown	27	may	676	6	-1	0	unknown	no
4499	45	blue-collar	divorced	primary	no	942	no	no	cellular	21	nov	362	1	-1	0	unknown	no
4500	38	admin.	married	secondary	no	4196	yes	no	cellular	12	may	193	2	-1	0	unknown	no
4501	34	management	married	tertiary	no	297	yes	no	cellular	26	aug	63	4	-1	0	unknown	no
4502	42	services	married	secondary	no	-91	yes	yes	cellular	5	feb	43	1	-1	0	unknown	no
4503	60	self-employed	married	primary	no	362	no	yes	cellular	29	jul	816	6	-1	0	unknown	yes
4504	42	blue-collar	single	secondary	no	1080	yes	yes	cellular	13	may	951	3	370	4	failure	yes
4505	32	admin.	single	secondary	no	620	yes	no	unknown	26	may	1234	3	-1	0	unknown	yes
4506	42	unemployed	divorced	tertiary	no	-166	no	no	cellular	29	aug	85	4	-1	0	unknown	no
4507	33	services	married	secondary	no	288	yes	no	cellular	17	apr	306	1	-1	0	unknown	no
4508	42	admin.	married	unknown	no	642	yes	yes	unknown	16	may	509	2	-1	0	unknown	no
4509	51	technician	married	tertiary	no	2506	no	no	cellular	30	nov	210	3	-1	0	unknown	no
4510	36	technician	divorced	secondary	no	566	yes	no	unknown	20	may	129	2	-1	0	unknown	no
4511	46	blue-collar	married	secondary	no	668	yes	no	unknown	15	may	1263	2	-1	0	unknown	yes
4512	40	blue-collar	married	secondary	no	1100	yes	no	unknown	29	may	660	2	-1	0	unknown	no
4513	49	blue-collar	married	secondary	no	322	no	no	cellular	14	aug	356	2	-1	0	unknown	no
4514	38	blue-collar	married	secondary	no	1205	yes	no	cellular	20	apr	45	4	153	1	failure	no
4515	32	services	single	secondary	no	473	yes	no	cellular	7	jul	624	5	-1	0	unknown	no
4516	33	services	married	secondary	no	-333	yes	no	cellular	30	jul	329	5	-1	0	unknown	no
4517	57	self-employed	married	tertiary	yes	-3313	yes	yes	unknown	9	may	153	1	-1	0	unknown	no
4518	57	technician	married	secondary	no	295	no	no	cellular	19	aug	151	11	-1	0	unknown	no
4519	28	blue-collar	married	secondary	no	1137	no	no	cellular	6	feb	129	4	211	3	other	no
4520	44	entrepreneur	single	tertiary	no	1136	yes	yes	cellular	3	apr	345	2	249	7	other	no

There are 4521 instances with 16 features and 1 class, making up a 4521 by 17 matrix.

One observation from this was that there are categorical features that are recorded as strings. This can cause value errors in SciKitLearn, printing an error message as follows

ValueError: could not convert string to float: unknown

Therefore one step of data preprocessing was to fix this. This was done with LabelEncoder in sklearn.preprocessing.

```
le = preprocessing.LabelEncoder()
for columns in bankData.columns:
    if bankData[columns].dtype == object:
        bankData[columns] = le.fit_transform(bankData[columns])
    else:
        pass
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	30	10	1	0	0	1787	0	0	0	19	10	79	1	-1	0	3	0
1	33	7	1	1	0	4789	1	1	0	11	8	220	1	339	4	0	0
2	35	4	2	2	0	1350	1	0	0	16	0	185	1	330	1	0	0
3	30	4	1	2	0	1476	1	1	2	3	6	199	4	-1	0	3	0
4	59	1	1	1	0	0	1	0	2	5	8	226	1	-1	0	3	0
5	35	4	2	2	0	747	0	0	0	23	3	141	2	176	3	0	0
6	36	6	1	2	0	307	1	0	0	14	8	341	1	330	2	1	0
7	39	9	1	1	0	147	1	0	0	6	8	151	2	-1	0	3	0
8	41	2	1	2	0	221	1	0	2	14	8	57	2	-1	0	3	0
9	43	7	1	0	0	-88	1	1	0	17	0	313	1	147	2	0	0
10	39	7	1	1	0	9374	1	0	2	20	8	273	1	-1	0	3	0
11	43	0	1	1	0	264	1	0	0	17	0	113	2	-1	0	3	0
12	36	9	1	2	0	1109	0	0	0	13	1	328	2	-1	0	3	0
13	20	8	2	1	0	502	0	0	0	30	0	261	1	-1	0	3	1
14	31	1	1	1	0	360	1	1	0	29	4	89	1	241	1	0	0
15	40	4	1	2	0	194	0	1	0	29	1	189	2	-1	0	3	0
16	56	9	1	1	0	4073	0	0	0	27	1	239	5	-1	0	3	0
17	37	0	2	2	0	2317	1	0	0	20	0	114	1	152	2	0	0
18	25	1	2	0	0	-221	1	0	2	23	8	250	1	-1	0	3	0
19	31	7	1	1	0	132	0	0	0	7	5	148	1	152	1	1	0
20	38	4	0	3	0	0	1	0	0	18	9	96	2	-1	0	3	0
21	42	4	0	2	0	16	0	0	0	19	9	140	3	-1	0	3	0
22	44	7	2	1	0	106	0	0	2	12	6	109	2	-1	0	3	0
23	44	2	1	1	0	93	0	0	0	7	5	125	2	-1	0	3	0
24	26	3	1	2	0	543	0	0	0	30	4	169	3	-1	0	3	0
25	41	4	1	2	0	5883	0	0	0	20	9	182	2	-1	0	3	0
26	55	1	1	0	0	627	1	0	2	5	8	247	1	-1	0	3	0
27	67	5	1	3	0	696	0	0	1	17	1	119	1	105	2	0	0
28	56	6	1	1	0	784	0	1	0	30	5	149	2	-1	0	3	0
29	53	0	1	1	0	105	0	1	0	21	1	74	2	-1	0	3	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
4491	35	1	2	1	0	0	1	0	0	16	0	169	1	-1	0	3	0
4492	32	9	2	1	0	309	1	1	0	16	0	346	1	234	3	0	0
4493	28	9	2	2	0	0	1	0	2	4	6	205	6	-1	0	3	0
4494	26	9	2	1	0	668	1	0	2	28	8	576	3	-1	0	3	1
4495	48	4	1	2	0	1175	1	0	1	18	9	1476	3	-1	0	3	0
4496	30	1	2	1	0	363	0	0	0	28	5	171	3	-1	0	3	0
4497	31	2	2	2	0	38	0	0	0	20	9	185	2	-1	0	3	0
4498	31	4	1	2	0	1183	1	0	2	27	8	676	6	-1	0	3	0
4499	45	1	0	0	0	942	0	0	0	21	9	362	1	-1	0	3	0
4500	38	0	1	1	0	4196	1	0	0	12	8	193	2	-1	0	3	0
4501	34	4	1	2	0	297	1	0	0	26	1	63	4	-1	0	3	0
4502	42	7	1	1	0	-91	1	1	0	5	3	43	1	-1	0	3	0
4503	60	6	1	0	0	362	0	1	0	29	5	816	6	-1	0	3	1
4504	42	1	2	1	0	1080	1	1	0	13	8	951	3	370	4	0	1
4505	32	0	2	1	0	620	1	0	2	26	8	1234	3	-1	0	3	1
4506	42	10	0	2	0	-166	0	0	0	29	1	85	4	-1	0	3	0
4507	33	7	1	1	0	288	1	0	0	17	0	306	1	-1	0	3	0
4508	42	0	1	3	0	642	1	1	2	16	8	509	2	-1	0	3	0
4509	51	9	1	2	0	2506	0	0	0	30	9	210	3	-1	0	3	0
4510	36	9	0	1	0	566	1	0	2	20	8	129	2	-1	0	3	0
4511	46	1	1	1	0	668	1	0	2	15	8	1263	2	-1	0	3	1
4512	40	1	1	1	0	1100	1	0	2	29	8	660	2	-1	0	3	0
4513	49	1	1	1	0	322	0	0	0	14	1	356	2	-1	0	3	0
4514	38	1	1	1	0	1205	1	0	0	20	0	45	4	153	1	0	0
4515	32	7	2	1	0	473	1	0	0	7	5	624	5	-1	0	3	0
4516	33	7	1	1	0	-333	1	0	0	30	5	329	5	-1	0	3	0
4517	57	6	1	2	1	-3313	1	1	2	9	8	153	1	-1	0	3	0
4518	57	9	1	1	0	295	0	0	0	19	1	151	11	-1	0	3	0
4519	28	1	1	1	0	1137	0	0	0	6	3	129	4	211	3	1	0
4520	44	2	2	2	0	1136	1	1	0	3	0	345	2	249	7	1	0

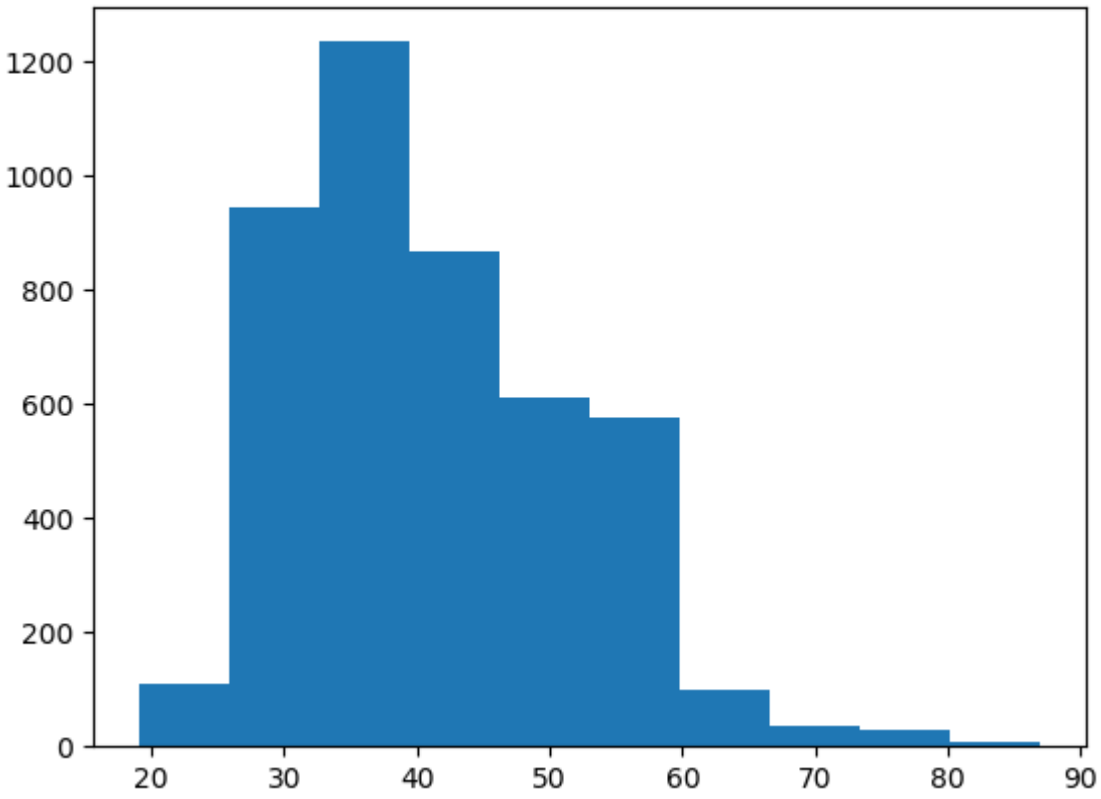
It is visible that categorical attributes are encoded into numerics.

Analysis on the data set

matplotlib.pyplot was used to plot data attributes, and to see the distribution of data.
For example, for the first attribute, which is age, a histogram was used to show the distribution of age.

```
tempArr=[]
for i in range(4521):
    tempArr.append(bankData.values[i][0])
plt.hist(tempArr)
plt.show()
```

And the result was as follows.



It shows a leftward skew. Other integer or floating point number attributes could be analyzed thus as well.

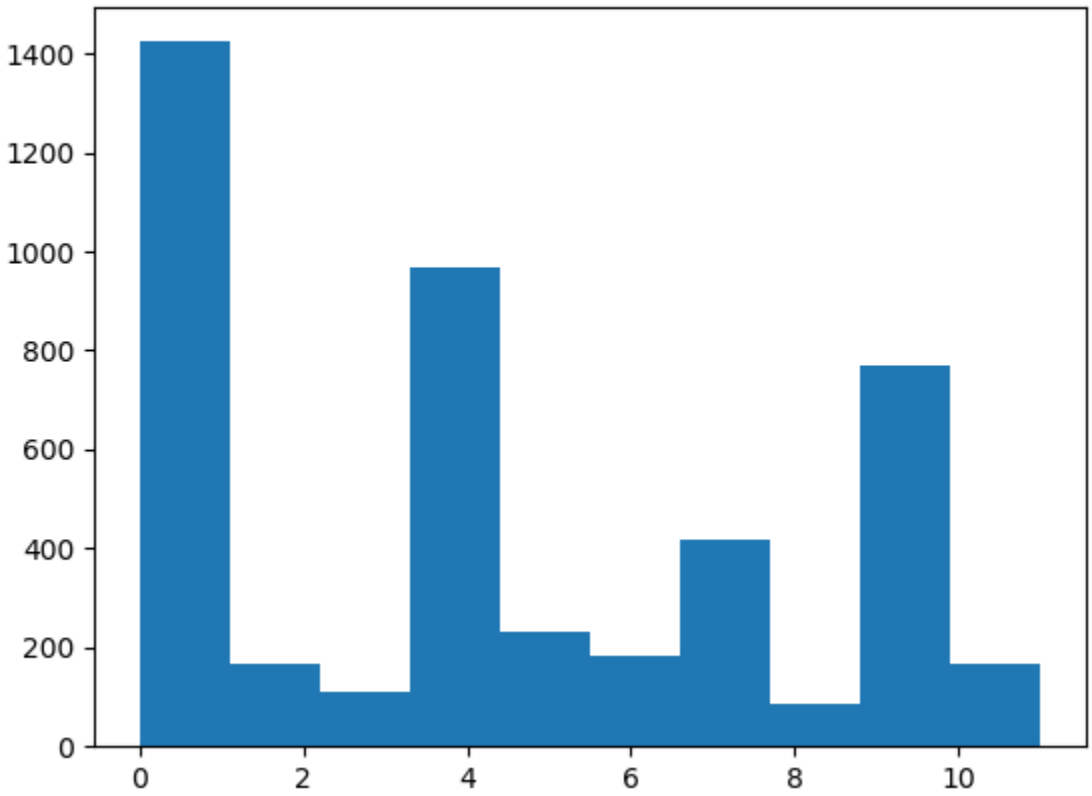
For categorical attributes, LabelEncoder was used to map categories into integers. Therefore, a way to show which category was mapped to which number is needed.

```
tempArr1 = ["admin.","unknown","unemployed","management","housemaid","entrepreneur","student", "blue-collar","self-employed","retired",
            "technician","services"]
tempArr1 = le.fit_transform(tempArr1)
print tempArr1
```

The result is

```
[ 0 11 10  4  3  2  8  1  6  5  9  7]
```

Now if you plot it using matplotlib.pyplot, you get



which shows that the most prevalent jobs were administrative, housemaid, and student. The distribution is obviously not fair, which may result in skewed results. Data preprocessing to even out the job distribution in the sample may be needed.

Also, there could be strong correlations between different attributes, including that between job and education, or loan and housing loan.

Also there were some attributes which could be deemed only weakly related to the outcome at best, such as the date on which the last contact was made. These attributes will need to be deprecated in the data preprocessing process.

sklearn.feature_selection.SelectPercentile was used to preprocess the data as follows.

```
bankDataArr = bankData.values
bankFeatures = []
bankTargets = []
for temp in bankDataArr:
    bankFeatures.append(temp[0:16])
    bankTargets.append(temp[16])

bankFeatures = SelectPercentile().fit_transform(bankFeatures, bankTargets)
```

Model construction

The data set was divided into a training set and a test set for cross validation. 20% of the data were reserved as test data, while the other 80% of the data were used as training data. Cross validation was used to prevent overfitting. The training set and the test set are chosen randomly, which ensures that it can be repeated for repeated hold out.

```
trainingData = []
trainingTargets = []
testData = []
testTargets = []
for i in range(4521):
    if(random.random() > 0.2):
        trainingData.append(bankFeatures[i])
        trainingTargets.append(bankTargets[i])
    else:
        testData.append(bankFeatures[i])
        testTargets.append(bankTargets[i])
```

Then the 4 models were trained on the training data, and then run on the test data.

```
clf = DecisionTreeClassifier().fit(trainingData, trainingTargets)
print clf.score(testData, testTargets)

logReg = linear_model.LogisticRegression()
logReg.fit(trainingData, trainingTargets)
print logReg.score(testData, testTargets)

gnb = GaussianNB()
gnb.fit(trainingData, trainingTargets)
print gnb.score(testData, testTargets)

mlp = MLPClassifier(hidden_layer_sizes=(16,15,14,13,12,11,10))
mlp.fit(trainingData, trainingTargets)
print mlp.score(testData, testTargets)
```

The results are as follows.

```
0.874468085106383
0.8872340425531915
0.8819148936170212
0.8840425531914894
```

It can be seen that although there was no data preprocessing other than encoding categorical data into numeric data, the classifier models are doing quite well, averaging higher than 0.85 in successful prediction rate. Also, it is visible that although these models don't differ vastly in their prediction accuracy, logistic regression did the best in terms of accuracy.

Data analysis from the models

Overall, logistic regression yielded the highest prediction rate, despite it was quite a close call. The results yielded from the models indicate that the data collected was useful in predicting whether the

recipient will subscribe to the marketed product or not, since the models yielded a very high prediction rate. Although this may seem to be the norm, it is not necessarily always the case, as can be seen in the next data set.

Dota2 data set

Dota2 is a game in which there are 5 players on each team. Each player picks a hero to play a certain role throughout the game. The Dota2 dataset recorded which heroes were used in which location, game mode, and game type, and whether the team won the game or not.

Each row of the dataset is a single game with the following features (in the order in the vector):
1. Team won the game (1 or -1)
2. Cluster ID (related to location)
3. Game mode (eg All Pick)
4. Game type (eg. Ranked)
5 - end: Each element is an indicator for a hero. Value of 1 indicates that a player from team '1' played as that hero and '-1' for the other team. Hero can be selected by only one player each game. This means that each row has five '1' and five '-1' values.

In other words, this data set can be used to determine not only which heroes were effective in which game modes, but also how the combination of heroes on one side and the other side affects the outcome of the game.

Data exploration

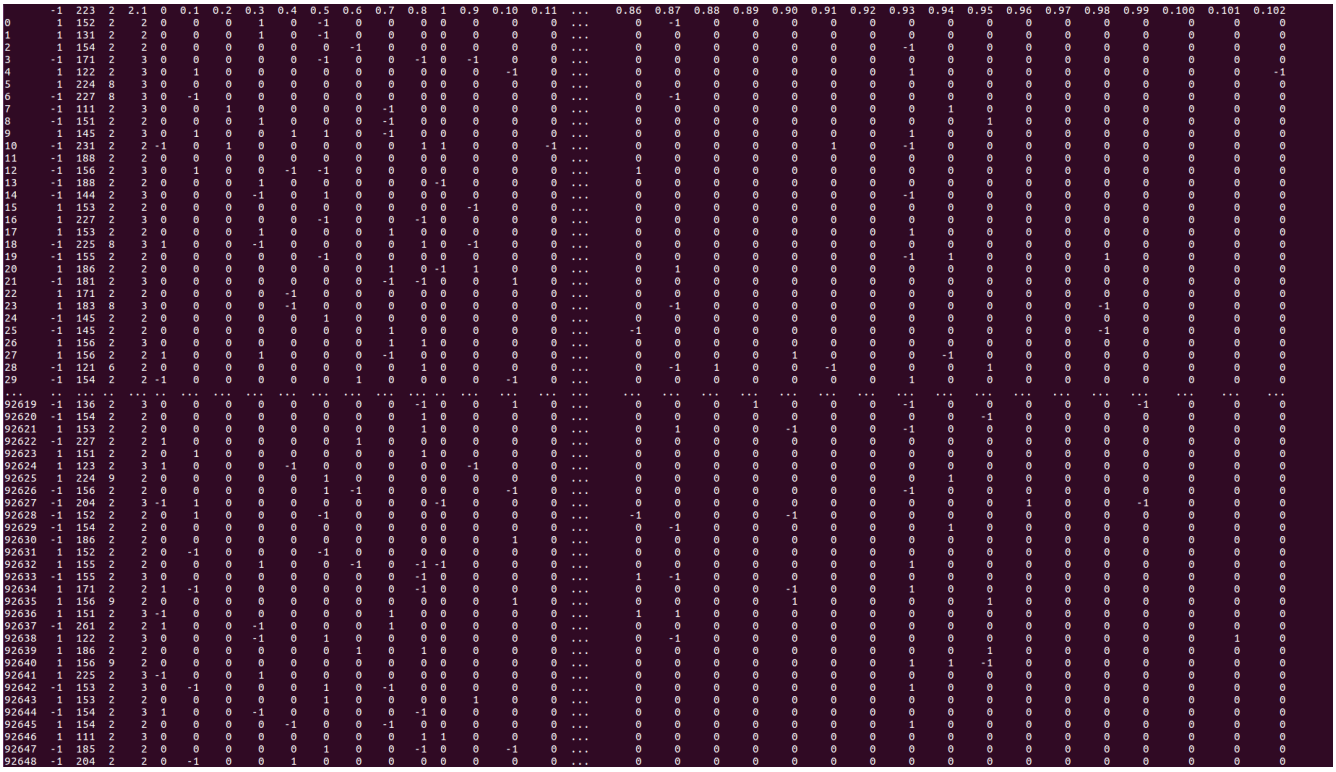
A similar procedure was conducted on the Dota2 data. First, the CSV data was loaded into python using pandas library.

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn import linear_model
from sklearn import preprocessing
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
import matplotlib.pyplot as plt

dotaData = pd.read_csv("/home/john/Desktop/AI/data/dota2Train.csv", sep=',')

print "\n\n\n"
print dotaData
```

The result was as follows



There are 92649 instances with 117 attributes.

Data analysis

The data set is very sparse, with the majority of values being 0. This is because only 10 heroes out of 113 can be selected in a single game, meaning that for every row, 103 elements out of 117 will be 0. This makes it harder to analyze the data through charts. Also, while being sparse, it can be seen that some columns are more filled than other columns, indicating that some heroes were picked more often that other heroes.

In this data set, there are no string values. To be more precise, the categorical attributes are already put into rows and changed into boolean values. The mapping is as follows.
https://github.com/kronusme/dota2-api/blob/master/data/heroes.json
Therefore, LabelEncoder was not needed for this data set.

Model construction

The data set was divided into a training set and a test set.

```
trainingData = []
trainingTargets = []
testData = []
testTargets = []
for i in range(len(dotaTargets)):
    if(random.random() > 0.2):
        trainingData.append(dotaFeatures[i])
        trainingTargets.append(dotaTargets[i])
    else:
        testData.append(dotaFeatures[i])
        testTargets.append(dotaTargets[i])
```

20% were used for test data, and the rest was used for training data. Training and test data sets were chosen at random to ensure repeatability, contributing to repeated hold out. Cross validation is needed to validate the model against overfitting.

Then, without any preprocessing or optimization, the classifier models were constructed and tested.

```
clf = DecisionTreeClassifier().fit(trainingData, trainingTargets)
print clf.score(testData, testTargets)

logReg = linear_model.LogisticRegression()
logReg.fit(trainingData, trainingTargets)
print logReg.score(testData, testTargets)

gnb = GaussianNB()
gnb.fit(trainingData, trainingTargets)
print gnb.score(testData, testTargets)

mlp = MLPClassifier(hidden_layer_sizes=(110, 50, 25, 12, 6, 3))
mlp.fit(trainingData, trainingTargets)
print mlp.score(testData, testTargets)
```

The result was as follows.

```
0.546
0.607
0.564
0.6055
```

The models did only slightly better than completely random prediction, which would yield around 0.5 accuracy, since there are only two classes. There can be three reasons for this. Either the classifier models were inadequately constructed, the data was inadequately preprocessed, or the game is very well balanced out so that the selection of heroes do not guarantee a win or a loss. At this point, the third reason seems to be the most appropriate, but other potential causes need to be checked.

In order to figure out whether cleaning the data would be helpful or not, feature selection was used. In particular, SelectPercentile was used from sklearn.feature_selection.

```
dotaDataArr = dotaData.values
dotaFeatures = []
dotaTargets = []
for temp in dotaDataArr:
    dotaFeatures.append(temp[1:117])
    dotaTargets.append(temp[0])

dotaFeatures = SelectPercentile().fit_transform(dotaFeatures, dotaTargets)

trainingData = dotaFeatures[0:90649]
trainingTargets = dotaTargets[0:90649]
testData = dotaFeatures[90649:92649]
testTargets = dotaTargets[90649:92649]
```

The result was as follows.

```
0.566
0.5775
0.5735
0.526
```

The accuracy did not improve. In fact, it got closer to 0.5, which is random accuracy given two outcomes.

This time, changes were made to the multi layered perceptron model to see if increasing hidden layers would improve the prediction accuracy of the model.

```
print("\n\n\n")
mlp = MLPClassifier(hidden_layer_sizes=(110, 50, 25, 12, 6, 3))
mlp.fit(trainingData, trainingTargets)
print mlp.score(testData, testTargets)

mlp1 = MLPClassifier(hidden_layer_sizes=(110, 75, 50, 40, 25, 20, 12, 9, 6, 4, 3))
mlp1.fit(trainingData, trainingTargets)
print mlp1.score(testData, testTargets)

mlp2 = MLPClassifier(hidden_layer_sizes=(110, 50, 25))
mlp2.fit(trainingData, trainingTargets)
print mlp2.score(testData, testTargets)
```

The result was as follows.

```
0.526
0.526
0.6065
```

Strangely, the model with the least layers yielded the highest accuracy. However, hardly any improvement was made on the model. This shows that the reason classifiers are doing poorly in predictions is most likely because the game is well balanced out and hero selection does not dictate win

or loss of the game. In that case, the model with the highest accuracy is ironically the least accurate, since predictions made from unrelated data are supposed to be haphazard.

Data analysis from the models

Prediction models constructed on the dota2 data set tells us that it’s hard to predict the outcome of a game based on the given features – region, game mode, game type, and champion selection. In other words, rather unrelated data were collected. This however is a positive sign for the game since it means that it’s difficult to win a game merely by choosing good heroes. In order to predict the outcomes, other attributes will need to be factored in, such as accumulated play time of each team, average rank of each player, or the age of the players. Overall, it shows that human factors may be more important in deciding the outcome of a game.

Conclusion

This experiment explored various ways to improve data exploitation through data preprocessing and model refinement. One interesting outcome was that cleaning the data of noise could actually bring down the prediction accuracy of models, if features are unrelated to classes. This makes sense, since it would be absurd to be able to yield high prediction rates with unrelated data. For example, if if your model predicts a basketball player’s 3 pointer success rate by the color of his socks with over 85% accuracy, it means one of two things. Either the color of socks does indeed contribute to 3 pointer success rates, or your model is constructed poorly. In such cases, the model needs to be refined or the data need to be processed.