Photo Source: Larryhw | Dreamstime.com

Campaign for Deposit Subscription

- Feature and Model Selections for Predictions

Presenter: Mingyuhui Liu (Jane)

Instructor: Joel Klein

Agenda

- Introduction
- Preprocessing
- SQL
- Feature and Model Selections
- Prediction

Introduction

- **Data:**
  - The dataset is from UCI (Sérgio Moro, Paulo Cortez and Paulo Rita, 2014.)
  - Bank Campaign data: whether the customer subscribed the deposit
- **Objective:**
  - Find best prediction model, and suggest strategies for bankers.
- **Methodologies:**
  - SQL
  - Handling categorical variables
  - PCA
  - SVC, Logistic, Random Forest and Naïve Bayes.
  - Recall, Precision, F2 score, ROC, etc.

Preprocessing:

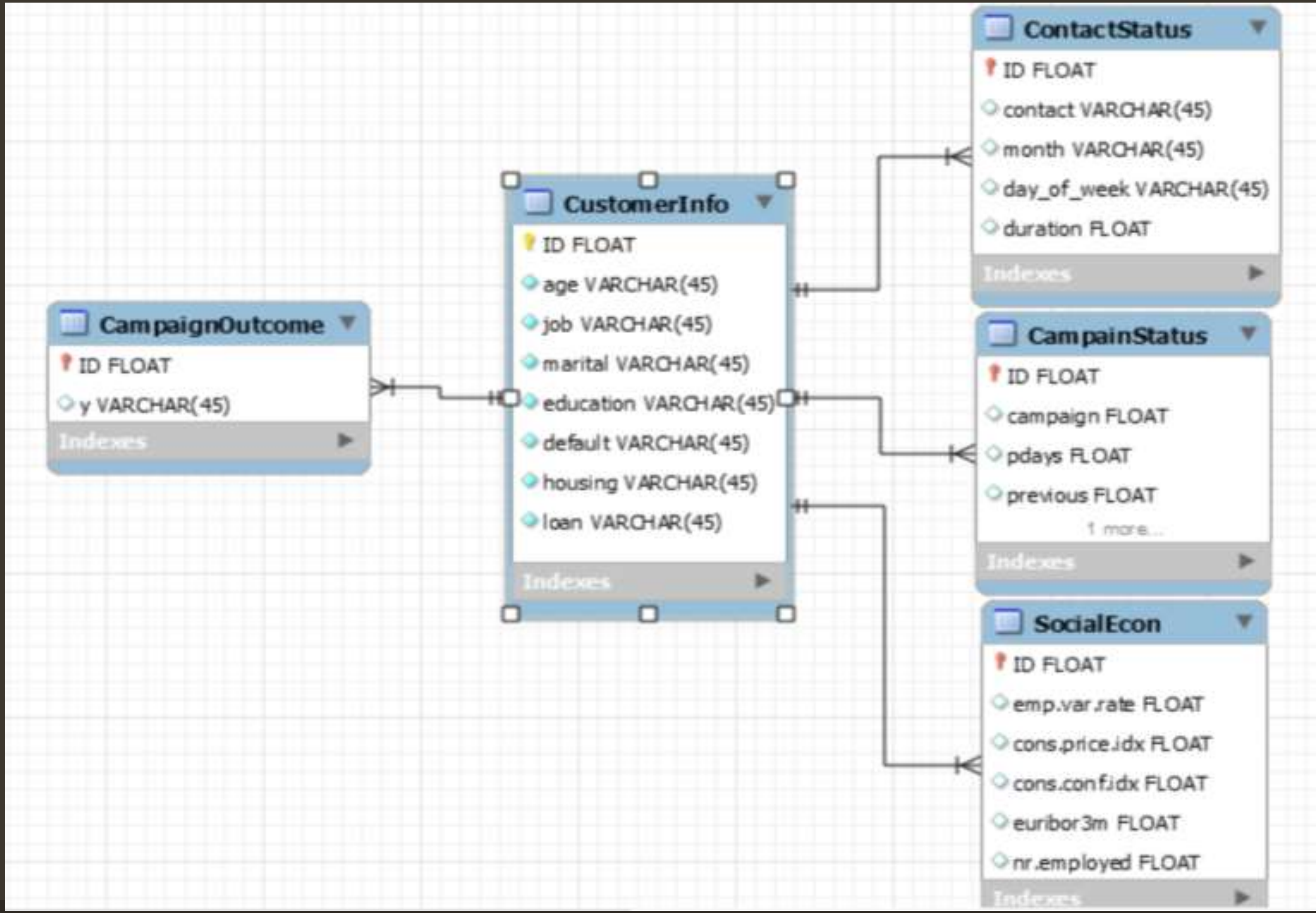
- Examples from the Original Data: 41188 instances, 23 columns

loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
yes	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no

- “client data”, “last contact of the current campaign”, “other attributes”, “social and economics attributes” and “output”

- Shuffle
- Add “ID” for creating SQL

SQL: Schema



CustomerInfo
ID FLOAT
age VARCHAR(45)
job VARCHAR(45)
marital VARCHAR(45)
education VARCHAR(45)
default VARCHAR(45)
housing VARCHAR(45)
loan VARCHAR(45)
Indexes
CampaignOutcome
ID FLOAT
y VARCHAR(45)
Indexes
ContactStatus
ID FLOAT
contact VARCHAR(45)
month VARCHAR(45)
day_of_week VARCHAR(45)
duration FLOAT
Indexes
CampainStatus
ID FLOAT
campaign FLOAT
pdays FLOAT
previous FLOAT
1 more...
Indexes
SocialEcon
ID FLOAT
emp.var.rate FLOAT
cons.price.idx FLOAT
cons.conf.idx FLOAT
euribor3m FLOAT
nr.employed FLOAT
Indexes

SQL: Creating Tables & Python

- Load into SQL;
- Merge into DataFrame

Handling Categorical Variables

- **Missing Values**
  - Stored as “Unknown” in some categorical data

- **Categorical Variables**
  - Categorical data includes:
    - 'job', 'marital', 'education’, etc.

  - Three ways to treat them:
    - Drop;
    - Ordinal concepts -> numerical;
    - One-hot-coding.

Handling Categorical Variables

- **Categorical Variables**
  - **Drop:**
    - job;
    - default
  - **Convert to numerical/continuous**
    - month
    - day_of_week
    - education
  - **One-hot-coding**
    - marital
    - housing
    - loan
    - etc

Handling Categorical Variables

- **Python:** categorical => numerical

```
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
```

No. Since I need full control of my encodings.

```
cleanup_1 = {"month": {"jan":1, "feb":2, "mar":3, "apr":4, "may":5, "jun":6, "jul":7, "aug": 8,
                       "sep":9, "oct":10, "nov":11, "dec":12},
             "day_of_week": {"mon": 1, "tue": 2, "wed": 3, "thu": 4, "fri": 5},
             "education": {"illiterate":0, "basic.4y":1, "basic.6y":2, "basic.9y":3,
                           "high.school":4, "professional.course":5, "university.degree":6},
             "y": {"no\r": 0, "yes\r": 1}}

#To convert the columns to numbers using replace :
obj_df.replace(cleanup_1, inplace=True)
obj_df.head()
```

	default	education	housing	job	loan	marital	y	poutcome	contact	day_of_week	month
0	no	6	yes	admin.	no\r	single	0	nonexistent\r	cellular	1	11
1	no	6	yes	entrepreneur	no\r	single	0	nonexistent\r	cellular	3	11
2	no	3	no	blue-collar	no\r	single	0	nonexistent\r	telephone	2	5
3	no	3	yes	self-employed	no\r	divorced	0	failure\r	cellular	3	4

Handling Categorical Variables

- **Python:** categorical => one-hot-code

DictVectorizer from SKLearn → No. Since I need full control of my encodings.

```python
obj_df = pd.get_dummies(obj_df, columns=["marital", "housing", "loan", "contact", "poutcome"])
```

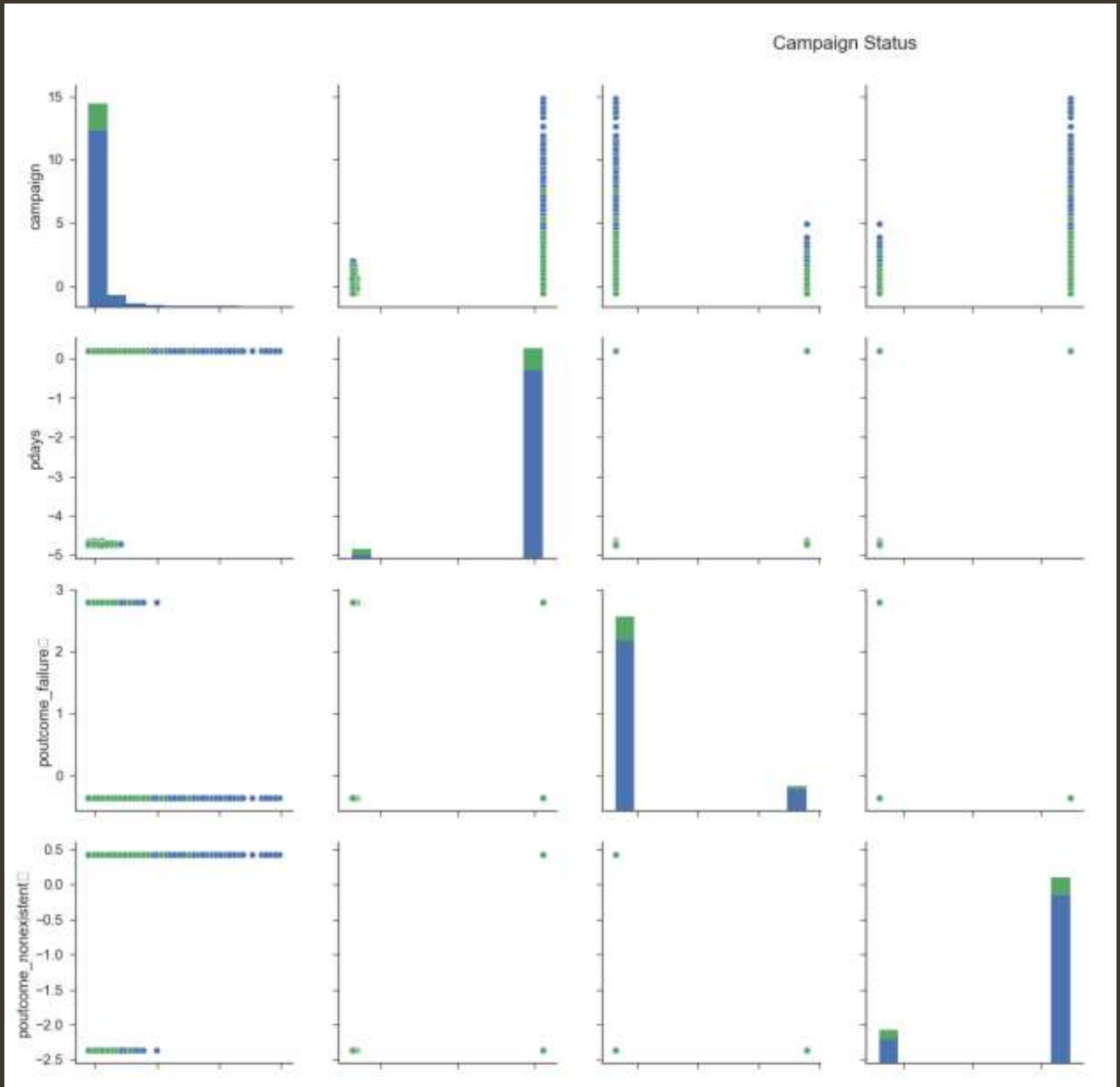
education	y	day_of_week	month	marital_divorced	marital_married	marital_single	housing_no	housing_yes	loan_no	loan_yes	contact_cellular
6	0	1	11	0	0	1	0	1	1	0	1
6	0	3	11	0	0	1	0	1	1	0	1
3	0	2	5	0	0	1	1	0	1	0	0
3	0	3	4	1	0	0	0	1	1	0	1
3	0	5	5	0	1	0	0	1	1	0	0
3	0	4	4	0	1	0	1	0	1	0	1
1	0	1	5	0	1	0	1	0	1	0	1

- **Standardize the data.**

Plotting

- **Seaborn pair plot**
  - Example: campaign status

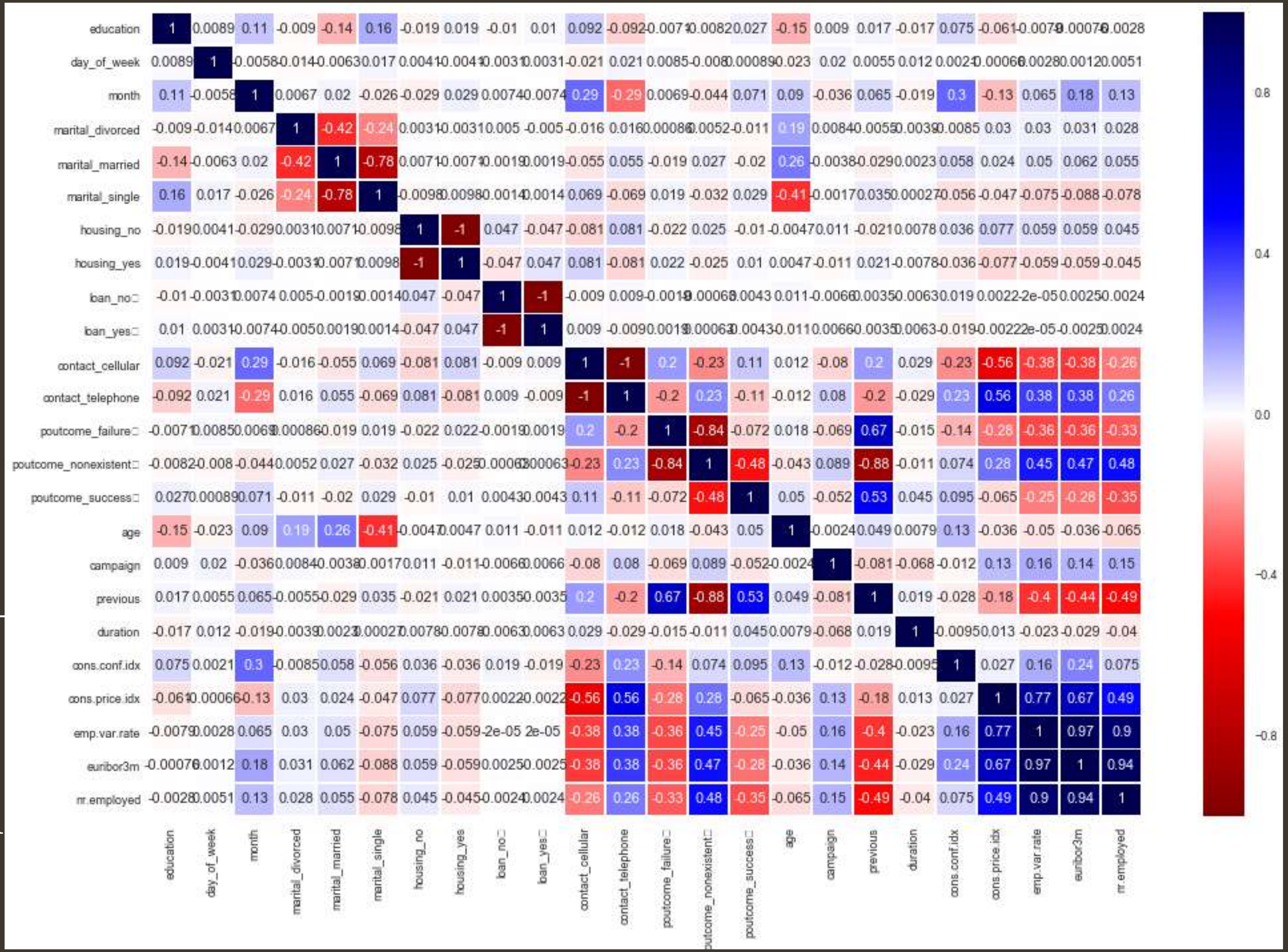
- **Conclusions:**
  - Highly biased on some variables;
  - Y itself is highly biased;



Plotting

- **Seaborn heatmap for correlations**

- **Drop correlated columns**: 'marital_single', 'housing_ 'loan_yes', 'contact_telephone', 'poutcome_nonexistent', 'previous', 'emp.var.rate', 'euribor3m', 'nr.employed



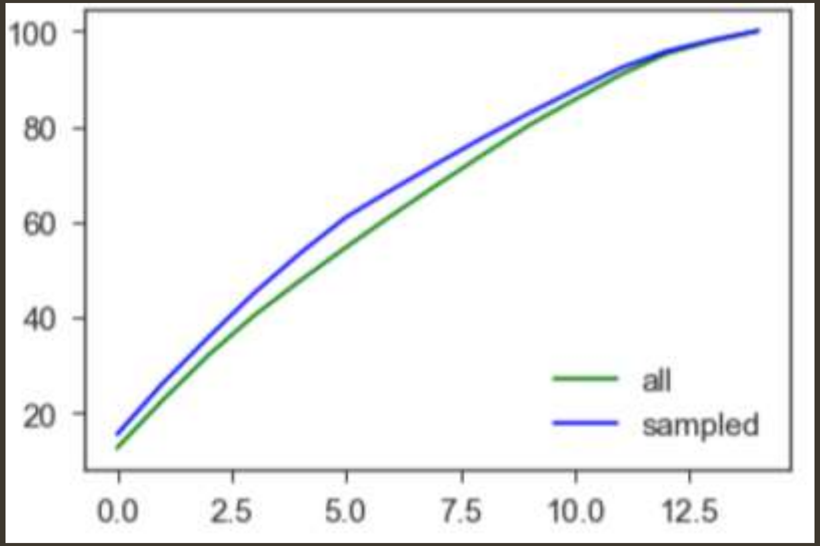
Feature Selection and Model Selection

- **Feature selections:**
  - Simple
  - PCA with all features
  - PCA with only numerical features

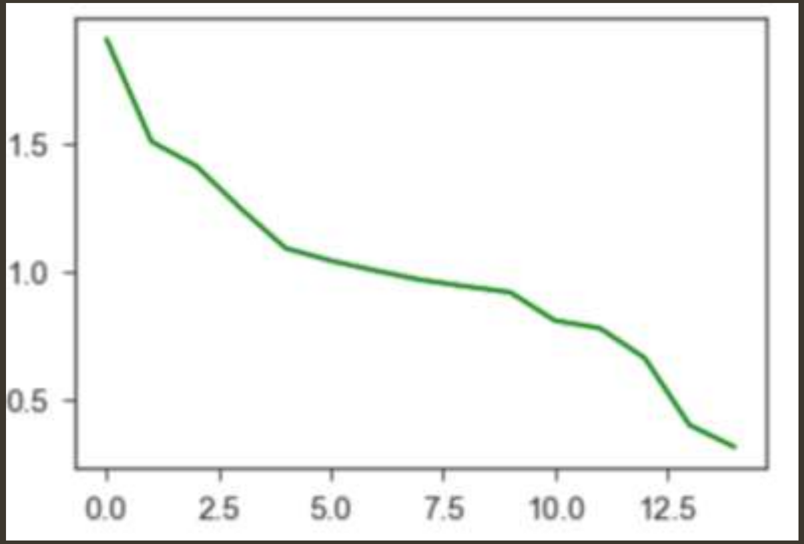
Feature Selection

- **PCA**: with all features
(All-data and down-sampled)

Variance covered



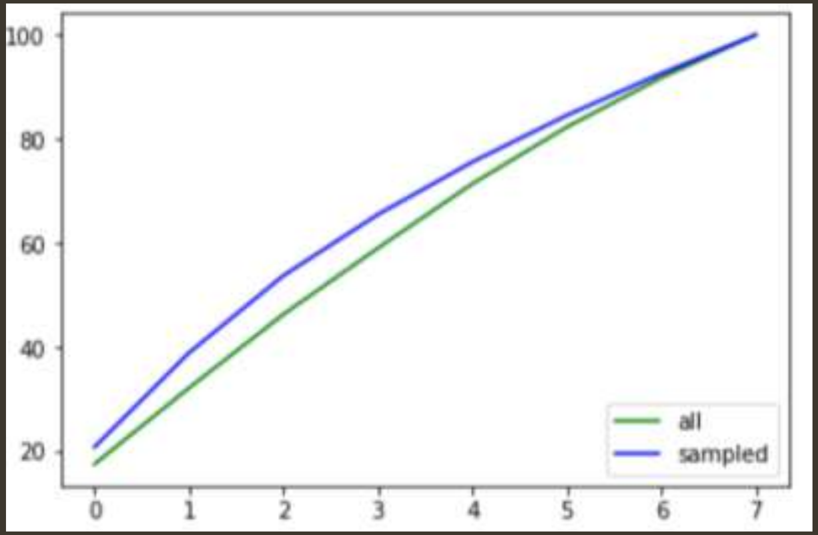
Scree plot



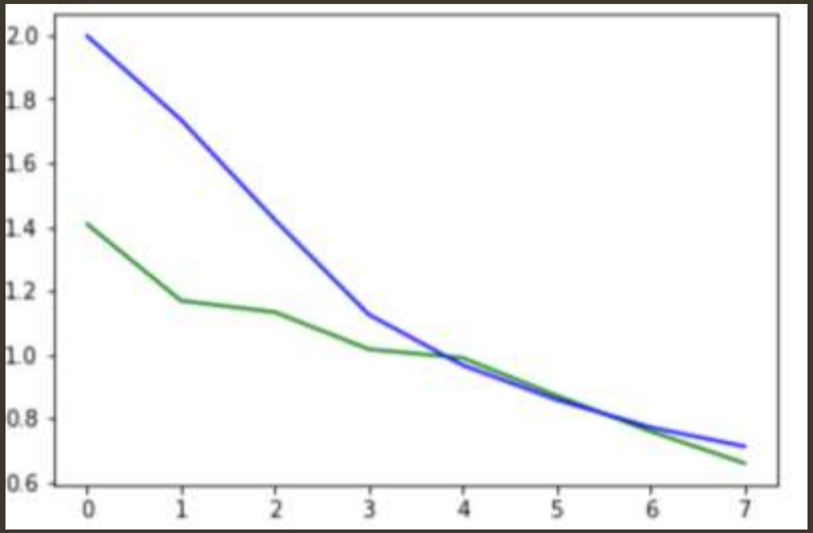
N =7;
Variance ~ 65%

Feature Selection

- **PCA**: with only numerical data, and then join with categorical data



Variance covered



Scree plot

N = 4
Variance ~ 60%

Model Selection: Models

- **All-data vs Down-sampled data (Equal classes)**
  - 3859 ‘no’ + 3859 ‘yes’

- **Models:**
  - LinearSVC
  - Logistic
  - Random Forest
  - Naïve Bayes

- **Training vs Testing size:**
  - 30%, 40%, 50%;
  - Control the random_state.

Model Selection: Evaluation Metrics

- **Accuracy**
- **Precision**
  - for ‘yes’
- **Recall**
  - For ‘yes’
- **F2 Score**
  - Beta = 2, so that recall is more important.
- **ROC/AUC**
  - Receiver operating characteristic curve
  - Area under the curve.

Model Selection: results

Features	Validation	LinearSVC		Logistic Classifier		Random Forest		Bernoulli NB	
		All Data	Balanced	All Data	Balanced	All Data	Balanced	All Data	Balanced
15 selected features manually	Accuracy Score	0.891	0.797	0.893	0.8	0.895	0.843	0.888	0.76
	Precision of 'yes'	0.61	0.83	0.61	0.82	0.6	0.84	0.64	0.78
	Recall of 'yes'	0.29	0.75	0.33	0.77	0.44	0.85	0.18	0.74
	F2 Score of 'yes'	0.324	0.767	0.36	0.783	0.464	0.862	0.205	0.744
	AUC*	0.879	0.881	0.878	0.881	0.886	0.815	N/A	
7 selected features after PCA	Accuracy Score	0.89	0.782	0.891	0.784	N/A		0.878	0.711
	Precision of 'yes'	0.61	0.83	0.61	0.82			0	0.69
	Recall of 'yes'	0.27	0.72	0.29	0.74			0	0.78
	F2 Score of 'yes'	0.301	0.742	0.322	0.755			0	0.758
	AUC*	0.858	0.868	0.859	0.755			N/A	
4 selected continuous + 7 Categorical	Accuracy Score	0.893	0.793	0.893	0.81			0.887	0.769
	Precision of 'yes'	0.68	0.82	0.68	0.84			0.65	0.76
	Recall of 'yes'	0.28	0.75	0.28	0.76			0.16	0.8
	F2 Score of 'yes'	0.314	0.768	0.314	0.774			0.192	0.794
	AUC*	0.851	0.875	0.851	0.885			N/A	

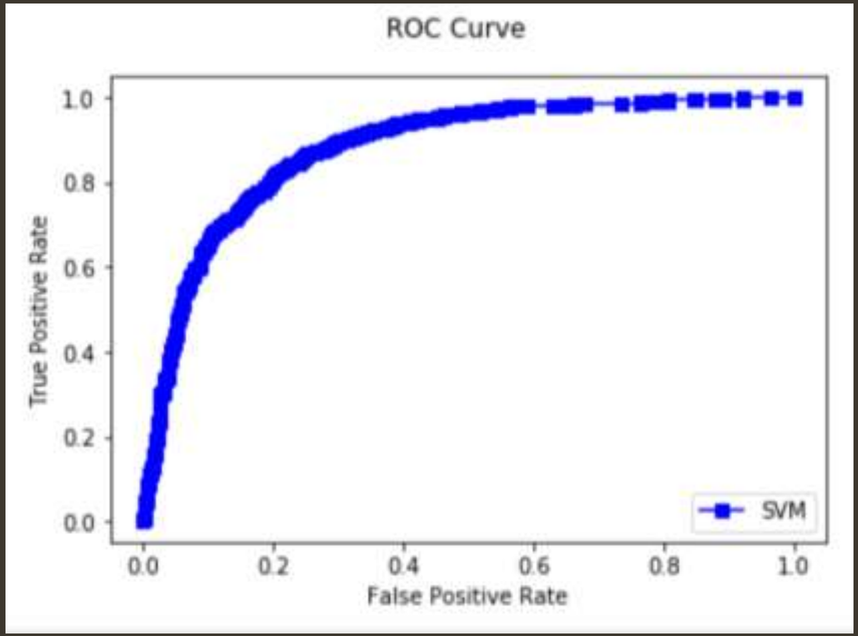
*: For Random Forest, this section will record the Out-of-bag accuracy scores.

Model Selection: LinearSVC

Summary:

- Balanced; 30% of Testings; Linear.

ROC: AUC = 0.881



Contingency Table: [[1011 163]
 [268 874]]

Model Selection: LinearSVC

Coefficients: top 10

- education 0.063
- month -0.078
- marital_married -0.072
- contact_cellular 0.146
- Loan_no 0.189
- age 0.067
- campaign -0.094
- duration 0.495
- cons.conf.idx 0.077
- cons.price.idx -0.103

Choose the clients who:

- Higher educated;
- No loans;
- Elder;
- First-time customer

And the managers should:

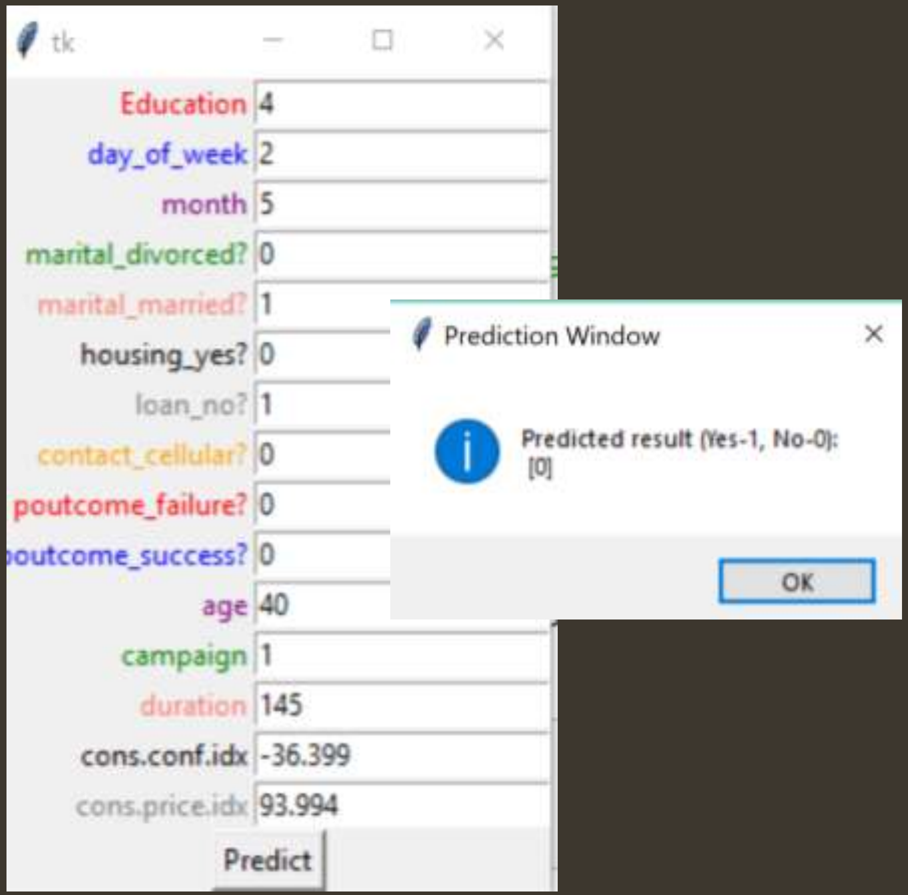
- Early months;
- Use cells;
- Hold the conversation;
- When consumer price index is low and consumer confidence is high.

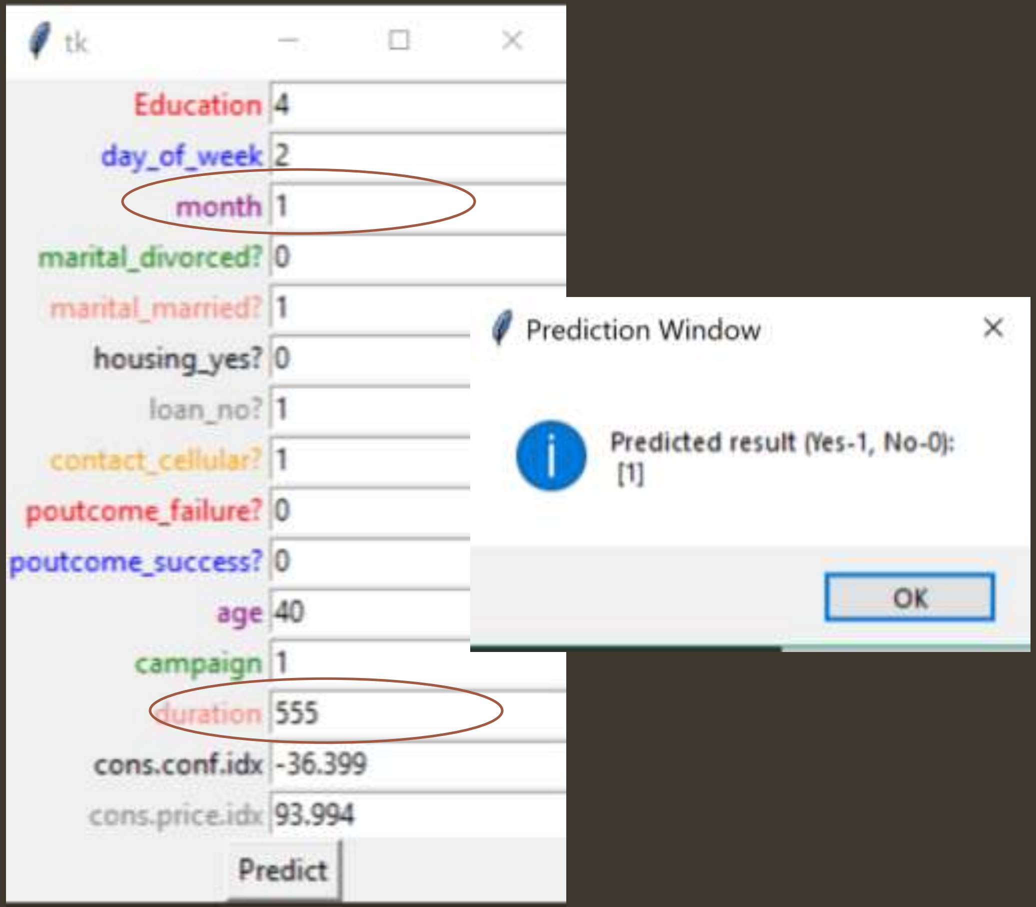
Prediction: Tackling the scaling problem

```
# Manually scale the input data.

std_original = df_original.std()
#print(std_original['education'])

mean_original=df_original.mean()
#print(mean_original)

col_needed = X_sampled.columns.tolist()

means = mean_original[col_needed].values
stds = std_original[col_needed].values
#print(stds.shape)

arr_original = np.array([4,2,5,0,1,0,1,0,0,0,40,1,145,-36.399, 93.994])
#[6,3,5,0,1,1,1,1,1,0,38,1,541,-46.2,92.893]
arr_scaled = (arr_original-means)/stds
arr = arr_scaled.reshape(1,-1)
#print(arr.shape)

print(arr)
```

```
[[-0.23041178 -0.69538014 -0.79868071 -0.36318842  0.8619419  -1.08757565
   0.43055191 -1.42656059 -0.35784484 -0.20162166  0.09386805 -0.55932607
  -0.43743926  0.8776455   0.80408191]]
```

Prediction: Simple GUI





Q&A

Thanks!

Results Discussion

- In this case:
  - “Yes” – 1
  - Focus more on Recall, using F2.
  - We want the bankers to focus more on the customers that are more likely to subscribe.

- Another way to interpret:
  - “No” – 1
  - Focus more on Precision, using F0.5.
  - We want the bankers to find out the main features that leads to non-subscription.

Model Selection: Random Forest

Summary:
- Balanced; 30% of Testings

Contingency Table:

[[1000 174]
 [147 995]]

Model Selection: Random Forest

Importance: Top 10 in order

- 'duration'
- 'cons.price.idx’
- 'age’
- 'cons.conf.idx’
- 'month'
- 'poutcome_success'
- 'day_of_week'
- 'campaign'
- 'education'
- 'contact_cellular'

```
digraph Tree {
node [shape=box] ;
0 [label="X[6] <= -0.946\ngini = 0.5\nsamples = 3434\nvalue = [2755, 2647]"] ;
1 [label="X[14] <= -1.119\ngini = 0.499\nsamples = 542\nvalue = [417, 450]"] ;
0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"] ;
2 [label="X[12] <= -0.428\ngini = 0.359\nsamples = 77\nvalue = [30, 98]"] ;
1 -> 2 ;
3 [label="X[8] <= 1.218\ngini = 0.5\nsamples = 21\nvalue = [16, 16]"] ;
2 -> 3 ;
4 [label="X[10] <= -1.406\ngini = 0.444\nsamples = 14\nvalue = [6, 12]"] ;
3 -> 4 ;
5 [label="gini = 0.0\nsamples = 2\nvalue = [0, 3]"] ;
4 -> 5 ;
6 [label="X[14] <= -2.107\ngini = 0.48\nsamples = 12\nvalue = [6, 9]"] ;
4 -> 6 ;
7 [label="gini = 0.0\nsamples = 3\nvalue = [4, 0]"] ;
6 -> 7 ;
8 [label="X[2] <= 0.607\ngini = 0.298\nsamples = 9\nvalue = [2, 9]"] ;
6 -> 8 ;
```

```sql
LOAD DATA LOCAL INFILE 'C:/Users/Jane Liu/Documents/GWU Spring2018/SQL/Final Project/CustomerInfo.csv'
INTO TABLE finalproject.customerinfo FIELDS TERMINATED BY ','
ENCLOSED BY '"' LINES TERMINATED BY '\n';

select * from finalproject.customerinfo;
```

esult Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

ID	age	job	marital	education	default	housing	loan
0	30	admin.	single	university.degree	no	yes	no
2	42	entrepreneur	single	university.degree	no	yes	no
3	28	blue-collar	single	basic.9y	no	no	no
4	42	technician	married	professional.course	unknown	yes	no
5	37	self-employed	divorced	basic.9y	no	yes	no
6	27	services	married	basic.9y	no	yes	no
7	59	blue-collar	married	basic.9y	no	no	no
8	28	blue-collar	married	basic.4y	no	no	no
9	38	admin.	married	university.degree	no	yes	no
10	34	blue-collar	married	basic.4y	no	yes	yes

tomerinfo 8

F-score

- Many applications require a balance between precision and recall.
- $F_\beta = \frac{(1+\beta^2) Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall}$
    - Recall is β times more important than precision
        - Van Rijsbergen, C. J. (1979). *Information Retrieval* (2nd ed.). Butterworth.
- When β=1 they are equally important. This is a widely-used balance between these two quantities
    - $F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$ -- the "harmonic mean" of precision and recall
- Other applications:
    - β=2 (F_2 score) recall is twice as important as precision
    - β=0.5 ($F_{0.5}$ score) precision is twice as important as recall