
# Machine Learning

## Winter 2020

## HW #4

**Jiyoon Chung**

**Arman Bhuiyan**

**Hikaru Sugimori**

**Deepak Putchakayala**

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score
```

In [6]:

```
x_df = pd.read_csv('X.csv').drop('Unnamed: 0', axis=1)
y_appetency_df = pd.read_csv('Y_appetency.csv').drop('Unnamed: 0', axis=1)
y_churn_df = pd.read_csv('Y_churn.csv').drop('Unnamed: 0', axis=1)
y_upselling_df = pd.read_csv('Y_upselling.csv').drop('Unnamed: 0', axis=1)
```

In [8]:

```
# look at the data
x_df
```

Out[8]:

| | Var1 | Var2 | Var3 | Var4 | Var5 | Var6 | Var7 | Var8 | Var9 | Var10 | ... | Var221 | Var222 | Var223 | Var224 | Var225 | Var226 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | NaN | NaN | NaN | NaN | NaN | 1526.0 | 7.0 | NaN | NaN | NaN | ... | oslk | fXVEsaq | jySVZNlOJy | NaN | NaN | xb3\ |
| 1 | NaN | NaN | NaN | NaN | NaN | 525.0 | 0.0 | NaN | NaN | NaN | ... | oslk | 2Kb5FSF | LM8l689qOp | NaN | NaN | fKCe |
| 2 | NaN | NaN | NaN | NaN | NaN | 5236.0 | 7.0 | NaN | NaN | NaN | ... | Al6ZaUT | NKv4yOc | jySVZNlOJy | NaN | kG3k | Qu4 |
| 3 | NaN | NaN | NaN | NaN | NaN | NaN | 0.0 | NaN | NaN | NaN | ... | oslk | CE7uk3u | LM8l689qOp | NaN | NaN | FSa2 |
| 4 | NaN | NaN | NaN | NaN | NaN | 1029.0 | 7.0 | NaN | NaN | NaN | ... | oslk | 1J2cvxe | LM8l689qOp | NaN | kG3k | FSa2 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 49995 | NaN | NaN | NaN | NaN | NaN | 357.0 | 0.0 | NaN | NaN | NaN | ... | oslk | EROH7Cg | LM8l689qOp | NaN | NaN | 7FJC |
| 49996 | NaN | NaN | NaN | NaN | NaN | 1078.0 | 0.0 | NaN | NaN | NaN | ... | oslk | GfSQowC | LM8l689qOp | NaN | kG3k | FSa2 |
| 49997 | NaN | NaN | NaN | NaN | NaN | 2807.0 | 7.0 | NaN | NaN | NaN | ... | oslk | dh6ql2t | LM8l689qOp | NaN | ELof | fKCe |
| 49998 | NaN | NaN | NaN | 0.0 | NaN | NaN | NaN | NaN | NaN | NaN | ... | oslk | 2fF2Oqu | LM8l689qOp | NaN | NaN | FSa2 |
| 49999 | NaN | NaN | NaN | NaN | NaN | 1694.0 | 7.0 | NaN | NaN | NaN | ... | oslk | IlvC99a | LM8l689qOp | NaN | NaN | xb3\ |

50000 rows × 230 columns

In [9]:

```
# Remove columns missing majority of data
# (i.e. columns containing >50% missing values (>25000))
x_df = x_df.dropna(thresh=25000, axis=1)
x_df
```

Out[9]:

| | Var6 | Var7 | Var13 | Var21 | Var22 | Var24 | Var25 | Var28 | Var35 | Var38 | ... | Var217 | Var218 | Var219 | Var220 | Var22 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1526.0 | 7.0 | 184.0 | 464.0 | 580.0 | 14.0 | 128.0 | 166.56 | 0.0 | 3570.0 | ... | sH5Z | cJvF | FzaX | 1YVfGrO | os |
| 1 | 525.0 | 0.0 | 0.0 | 168.0 | 210.0 | 2.0 | 24.0 | 353.52 | 0.0 | 4764966.0 | ... | NaN | NaN | FzaX | 0AJo2f2 | os |
| 2 | 5236.0 | 7.0 | 904.0 | 1212.0 | 1515.0 | 26.0 | 816.0 | 220.08 | 0.0 | 5883894.0 | ... | bHR7 | UYBR | FzaX | JFM1BiF | Al6ZaU |
| 3 | NaN | 0.0 | 0.0 | NaN | 0.0 | NaN | 0.0 | 22.08 | 0.0 | 0.0 | ... | eKej | UYBR | FzaX | L91Kliz | os |
| 4 | 1029.0 | 7.0 | 3216.0 | 64.0 | 80.0 | 4.0 | 64.0 | 200.00 | 0.0 | 0.0 | ... | H3p7 | UYBR | FzaX | OrnLfvc | os |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 49995 | 357.0 | 0.0 | 0.0 | 132.0 | 165.0 | 2.0 | 0.0 | 288.08 | 0.0 | 6042420.0 | ... | XXsx | cJvF | FzaX | 3JmRJnY | os |
| 49996 | 1078.0 | 0.0 | 2736.0 | 380.0 | 475.0 | 2.0 | 88.0 | 166.56 | 0.0 | 0.0 | ... | 4a9J | UYBR | FzaX | MMTv4zN | os |
| 49997 | 2807.0 | 7.0 | 1460.0 | 568.0 | 710.0 | 4.0 | 328.0 | 166.56 | 0.0 | 42210.0 | ... | DV70 | UYBR | FzaX | FM28hdx | os |
| 49998 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | 8Mfr | UYBR | FzaX | BV9YlW4 | os |
| 49999 | 1694.0 | 7.0 | 828.0 | 192.0 | 240.0 | 0.0 | 48.0 | 220.08 | 5.0 | 2691228.0 | ... | k0HX | cJvF | FzaX | ot6oLzk | os |

50000 rows × 69 columns

**we reduced from 230 columns to 69 columns**

In [10]:

```
# as suggested replace numerical missing values with the mean of observed values
x_df = x_df.fillna(x_df.mean())
x_df
```

Out[10]:

| | Var6 | Var7 | Var13 | Var21 | Var22 | Var24 | Var25 | Var28 | Var35 | Var38 | ... | V |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1526.000000 | 7.000000 | 184.000000 | 464.000000 | 580.000000 | 14.000000 | 128.00000 | 166.560000 | 0.00000 | 3.570000e+03 | ... | |
| 1 | 525.000000 | 0.000000 | 0.000000 | 168.000000 | 210.000000 | 2.000000 | 24.00000 | 353.520000 | 0.00000 | 4.764966e+06 | ... | |
| 2 | 5236.000000 | 7.000000 | 904.000000 | 1212.000000 | 1515.000000 | 26.000000 | 816.00000 | 220.080000 | 0.00000 | 5.883894e+06 | ... | |
| 3 | 1326.437116 | 0.000000 | 0.000000 | 234.518225 | 0.000000 | 4.507926 | 0.00000 | 22.080000 | 0.00000 | 0.000000e+00 | ... | |
| 4 | 1029.000000 | 7.000000 | 3216.000000 | 64.000000 | 80.000000 | 4.000000 | 64.00000 | 200.000000 | 0.00000 | 0.000000e+00 | ... | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 49995 | 357.000000 | 0.000000 | 0.000000 | 132.000000 | 165.000000 | 2.000000 | 0.00000 | 288.080000 | 0.00000 | 6.042420e+06 | ... | |
| 49996 | 1078.000000 | 0.000000 | 2736.000000 | 380.000000 | 475.000000 | 2.000000 | 88.00000 | 166.560000 | 0.00000 | 0.000000e+00 | ... | |
| 49997 | 2807.000000 | 7.000000 | 1460.000000 | 568.000000 | 710.000000 | 4.000000 | 328.00000 | 166.560000 | 0.00000 | 4.221000e+04 | ... | |
| 49998 | 1326.437116 | 6.809496 | 1249.688401 | 234.518225 | 290.245382 | 4.507926 | 96.82701 | 224.507669 | 0.71681 | 2.579107e+06 | ... | |
| 49999 | 1694.000000 | 7.000000 | 828.000000 | 192.000000 | 240.000000 | 0.000000 | 48.00000 | 220.080000 | 5.00000 | 2.691228e+06 | ... | |

50000 rows × 69 columns

In [11]:

```
# Replace categorical missing value with its mode
x_df = x_df.fillna(x_df.mode().iloc[0])
x_df.head()
```

Out[11]:

| | Var6 | Var7 | Var13 | Var21 | Var22 | Var24 | Var25 | Var28 | Var35 | Var38 | ... | Var217 | Var218 | Var219 | Var22 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1526.000000 | 7.0 | 184.0 | 464.000000 | 580.0 | 14.000000 | 128.0 | 166.56 | 0.0 | 3570.0 | ... | sH5Z | cJvF | FzaX | 1YVfGr |
| 1 | 525.000000 | 0.0 | 0.0 | 168.000000 | 210.0 | 2.000000 | 24.0 | 353.52 | 0.0 | 4764966.0 | ... | gvA6 | cJvF | FzaX | 0AJo2 |
| 2 | 5236.000000 | 7.0 | 904.0 | 1212.000000 | 1515.0 | 26.000000 | 816.0 | 220.08 | 0.0 | 5883894.0 | ... | bHR7 | UYBR | FzaX | JFM1Bi |
| 3 | 1326.437116 | 0.0 | 0.0 | 234.518225 | 0.0 | 4.507926 | 0.0 | 22.08 | 0.0 | 0.0 | ... | eKej | UYBR | FzaX | L91Kl |
| 4 | 1029.000000 | 7.0 | 3216.0 | 64.000000 | 80.0 | 4.000000 | 64.0 | 200.00 | 0.0 | 0.0 | ... | H3p7 | UYBR | FzaX | OrnLf |

5 rows × 69 columns

### convert the categorical variables into numbers to ensure classification machine learning models work

In [12]:

```
from sklearn.preprocessing import LabelEncoder

object_columns = list(x_df.select_dtypes(include=['object']).columns)
le = preprocessing.LabelEncoder()

for c in object_columns:
    x_df[c] = le.fit_transform(x_df[c])

x_df.head()
```

Out[12]:

| | Var6 | Var7 | Var13 | Var21 | Var22 | Var24 | Var25 | Var28 | Var35 | Var38 | ... | Var217 | Var218 | Var219 | Var220 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1526.000000 | 7.0 | 184.0 | 464.000000 | 580.0 | 14.000000 | 128.0 | 166.56 | 0.0 | 3570.0 | ... | 12236 | 1 | 11 | 94 |
| 1 | 525.000000 | 0.0 | 0.0 | 168.000000 | 210.0 | 2.000000 | 24.0 | 353.52 | 0.0 | 4764966.0 | ... | 9781 | 1 | 11 | 10 |
| 2 | 5236.000000 | 7.0 | 904.0 | 1212.000000 | 1515.0 | 26.000000 | 816.0 | 220.08 | 0.0 | 5883894.0 | ... | 8473 | 0 | 11 | 1293 |
| 3 | 1326.437116 | 0.0 | 0.0 | 234.518225 | 0.0 | 4.507926 | 0.0 | 22.08 | 0.0 | 0.0 | ... | 9161 | 0 | 11 | 1444 |
| 4 | 1029.000000 | 7.0 | 3216.0 | 64.000000 | 80.0 | 4.000000 | 64.0 | 200.00 | 0.0 | 0.0 | ... | 3834 | 0 | 11 | 1731 |

5 rows × 69 columns

### feature/variable selection

In [22]:

```
## convert the data frame to an array with out indexes
y_appetency = y_appetency_df.as_matrix().ravel()
```

```
/Users/deepakputchakayala/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2: FutureW
arning: Method .as_matrix will be removed in a future version. Use .values instead.
```

In [23]:

```
y_appetency
```

Out[23]:

```
array([-1, -1, -1, ..., -1, -1, -1])
```

In [24]:

```
## use random forest classifier

from sklearn.ensemble import RandomForestClassifier
forest = RandomForestClassifier(random_state=0, n_estimators=50).fit(x_df, y_appetency)
```

### get the feature scores of all of the variables

In [25]:

```
var_names = list(x_df)
var_scores = forest.feature_importances_
```

In [26]:

```
var_scores
```

Out[26]:

```
array([0.02085378, 0.00660307, 0.01673791, 0.01556624, 0.01697321,
       0.00792824, 0.0138639 , 0.02041241, 0.00420301, 0.01931318,
       0.00314434, 0.03391433, 0.0060191 , 0.0070666 , 0.02104379,
       0.0121004 , 0.01748339, 0.00393499, 0.02445564, 0.01185783,
       0.00909445, 0.01893513, 0.01166665, 0.01196942, 0.03509166,
       0.01921642, 0.01503358, 0.01833698, 0.04124293, 0.00542152,
       0.02185926, 0.02077975, 0.01600123, 0.00132292, 0.0088556 ,
       0.01275546, 0.02389609, 0.01500562, 0.01467752, 0.00084175,
       0.00221583, 0.02722537, 0.00566732, 0.00206828, 0.00134904,
       0.02347294, 0.03150008, 0.03424952, 0.03449705, 0.00313001,
       0.02279278, 0.00715059, 0.01014339, 0.00362896, 0.00223285,
       0.00066059, 0.00454081, 0.00719255, 0.02250812, 0.02931005,
       0.01002841, 0.0071738 , 0.02986279, 0.00433177, 0.03255649,
       0.0033407 , 0.01950584, 0.00463162, 0.00755317])
```

### order variable names to their scores

In [28]:

```
var_selection = dict(zip(var_names, var_scores))

from collections import OrderedDict
var_selection = OrderedDict(sorted(var_selection.items(), key=lambda t: t[1]))
```

In [29]:

```
var_selection
```

Out[29]:

```
OrderedDict([('Var210', 0.0006605899751529229),
             ('Var173', 0.0008417491551287499),
             ('Var143', 0.0013229166445293222),
             ('Var196', 0.0013490383766777496),
             ('Var195', 0.00206828355592741),
             ('Var181', 0.0022158250863581965),
             ('Var208', 0.00223284653581144),
             ('Var203', 0.003130014539512072),
             ('Var44', 0.003144338684970231),
             ('Var223', 0.0033406990050013157),
             ('Var207', 0.003628956454699797),
             ('Var78', 0.003934985891648502),
             ('Var35', 0.004203012016210799),
             ('Var221', 0.004331765919437885),
             ('Var211', 0.004540810110385476),
             ('Var227', 0.004631621667311638),
             ('Var132', 0.005421523146533028),
             ('Var193', 0.005667322805749791),
             ('Var65', 0.006019100547839883),
             ('Var7', 0.0066030714554311766),
             ('Var72', 0.007066597681661176),
             ('Var205', 0.007150587175351),
             ('Var219', 0.007173800700114202),
             ('Var212', 0.007192553707665614),
             ('Var228', 0.007553165633174522),
             ('Var24', 0.007928242303102344),
             ('Var144', 0.008855604926172704),
             ('Var85', 0.009094447011352565),
             ('Var218', 0.010028409436068288),
             ('Var206', 0.010143388627121198),
             ('Var109', 0.01166664671001927),
             ('Var83', 0.011857828500630153),
             ('Var112', 0.011969419805967028),
             ('Var74', 0.012100395521908471),
             ('Var149', 0.012755458605685506),
             ('Var25', 0.013863897907537287),
             ('Var163', 0.0146775245266567),
             ('Var160', 0.01500562422001022),
             ('Var123', 0.01503357528050354),
             ('Var21', 0.015566235134356378),
             ('Var140', 0.01600123294685374),
             ('Var13', 0.016737906188259245),
             ('Var22', 0.016973214955040034),
             ('Var76', 0.01748339308352511),
             ('Var125', 0.01833697656536476),
             ('Var94', 0.018935133628706722),
             ('Var119', 0.019216420582373824),
             ('Var38', 0.01931317547021065),
             ('Var226', 0.019505837797750593),
             ('Var28', 0.020412408160774467),
             ('Var134', 0.02077975256948976),
             ('Var6', 0.020853782522534475),
             ('Var73', 0.021043794248153454),
             ('Var133', 0.021859258047668875),
             ('Var216', 0.02250811780925778),
             ('Var204', 0.022792779467008292),
             ('Var197', 0.023472936650561636),
             ('Var153', 0.023896090398390645),
             ('Var81', 0.02445563744131191),
             ('Var192', 0.027225373454280853),
             ('Var217', 0.02931005055689533),
             ('Var220', 0.029862793775187173),
             ('Var198', 0.031500078238782106),
             ('Var222', 0.03255648846077877),
             ('Var57', 0.03391433187587417),
             ('Var199', 0.034249523304448026),
             ('Var202', 0.03449705191510502),
             ('Var113', 0.03509165705444437),
             ('Var126', 0.0412429278415927)])
```

### drop non-significant variables with 0.025 as cut off

In [31]:

```
dropping_vars = []

for key in var_selection:
    if var_selection.get(key) < 0.025:
        dropping_vars.append(key)
```

### keep only selected variables and create a new dataframe

In [32]:

```
x_selected_df = x_df.drop(dropping_vars, axis=1)

appetency_vars = list(x_selected_df.columns)

x_selected_df
```

Out[32]:

| | Var57 | Var113 | Var126 | Var192 | Var198 | Var199 | Var202 | Var217 | Var220 | Var222 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 4.076907 | 117625.60 | 8.00000 | 225 | 3302 | 4015 | 3682 | 12236 | 94 | 2964 |
| 1 | 5.408032 | -356411.60 | -0.55388 | 91 | 3697 | 4890 | 4422 | 9781 | 10 | 125 |
| 2 | 6.599658 | 405104.00 | -28.00000 | 248 | 1686 | 4900 | 448 | 8473 | 1293 | 1610 |
| 3 | 1.988250 | -275703.60 | -14.00000 | 269 | 2575 | 362 | 2953 | 9161 | 1444 | 817 |
| 4 | 4.552446 | 10714.84 | 58.00000 | 131 | 971 | 2399 | 5511 | 3834 | 1731 | 65 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 49995 | 2.757958 | 85899.60 | -28.00000 | 341 | 560 | 3502 | 3489 | 7428 | 257 | 969 |
| 49996 | 0.594958 | -1461768.00 | -30.00000 | 179 | 3269 | 596 | 2353 | 1060 | 1512 | 1125 |
| 49997 | 6.574023 | 105957.60 | -0.55388 | 326 | 1491 | 3705 | 2692 | 3040 | 1041 | 2834 |
| 49998 | 6.562059 | -345201.60 | -16.00000 | 115 | 2237 | 1770 | 1652 | 1874 | 837 | 142 |
| 49999 | 1.134800 | 150766.40 | -30.00000 | 225 | 453 | 2096 | 3736 | 10438 | 3621 | 1262 |

50000 rows × 10 columns

### split to test/train data

In [33]:

```
x_train, x_test = train_test_split(x_selected_df,
                 test_size=0.20, random_state=0)
y_appetency_train, y_appetency_test = train_test_split(y_appetency_df,
                 test_size=0.20, random_state=0)
```

### use gridserachCV

In [35]:

```
from sklearn.ensemble import RandomForestClassifier

grid = GridSearchCV(RandomForestClassifier(n_estimators=20), param_grid={'max_leaf_nodes': np.arange(100,2000,300
)}, cv = 5)
```

In [36]:

```
grid.fit(x_train, y_appetency_train.as_matrix().ravel())
```

```
/Users/deepakputchakayala/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1: FutureW
arning: Method .as_matrix will be removed in a future version. Use .values instead.
  """Entry point for launching an IPython kernel.
```

Out[36]:

```
GridSearchCV(cv=5, error_score='raise-deprecating',
             estimator=RandomForestClassifier(bootstrap=True, class_weight=None,
                                              criterion='gini', max_depth=None,
                                              max_features='auto',
                                              max_leaf_nodes=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              n_estimators=20, n_jobs=None,
                                              oob_score=False,
                                              random_state=None, verbose=0,
                                              warm_start=False),
             iid='warn', n_jobs=None,
             param_grid={'max_leaf_nodes': array([ 100,  400,  700, 1000, 1300, 1600, 1900])},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

In [37]:

```
print('Optimal leaf nodes', grid.best_params_['max_leaf_nodes'])
```

```
Optimal leaf nodes 1900
```

In [38]:

```
print('Accuracy', max(grid.cv_results_['mean_test_score']))
```

```
Accuracy 0.9823
```

### test the model on the test data set

In [41]:

```
forest_test = RandomForestClassifier(random_state=0, n_estimators=20, max_leaf_nodes=1900).fit(x_train, y_appeten
cy_train.as_matrix().ravel())

y_true = y_appetency_test
y_pred = forest_test.predict(x_test)
score = accuracy_score(y_true, y_pred)

print('appetency: ', score)
```

```
/Users/deepakputchakayala/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1: FutureW
arning: Method .as_matrix will be removed in a future version. Use .values instead.
  """Entry point for launching an IPython kernel.

appetency:  0.9816
```

In [ ]:

```

```