Introduction to Data Science

Homework 2 ¶

Student Name: Monil Shah

Student Netid: mds747

Preparing a Training Set and Training a Decision Tree

This is a hands-on task where we build a predictive model using Decision Trees discussed in class. For this part, we will be using the data in `cell2cell_data.csv`.

These historical data consist of 39,859 customers: 19,901 customers that churned (i.e., left the company) and 19,958 that did not churn (see the `"churndep"` variable). Here are the data set's 11 possible predictor variables for churning behavior:

```
Pos.  Var. Name  Var. Description
----- ---------- -------------------------------------------------
-----------
1     revenue    Mean monthly revenue in dollars
2     outcalls   Mean number of outbound voice calls
3     incalls    Mean number of inbound voice calls
4     months     Months in Service
5     eqpdays    Number of days the customer has had his/her current
 equipment
6     webcap     Handset is web capable
7     marryyes   Married (1=Yes; 0=No)
8     travel     Has traveled to non-US country (1=Yes; 0=No)
9     pcown      Owns a personal computer (1=Yes; 0=No)
10    creditcd   Possesses a credit card (1=Yes; 0=No)
11    retcalls   Number of calls previously made to retention team
```

The 12th column, the dependent variable `"churndep"`, equals 1 if the customer churned, and 0 otherwise.

1. Load the data and prepare it for modeling. Note that the features are already processed for you, so the only thing needed here is split the data into training and testing. Use pandas to create two data frames: train_df and test_df, where train_df has 80% of the data chosen uniformly at random without replacement (test_df should have the other 20%). Also, make sure to write your own code to do the splits. You may use any random() function numpy but DO NOT use the data splitting functions from Sklearn.

In [12]:
```
import pandas as pd
import numpy as np
df = pd.read_csv('/Users/monilshah/DataScienceCourse/ipython/data/cell2cell_
# Code here
train_split = 0.8
train = df.sample(frac=0.8)
test = df[~df.index.isin(train.index)]
print("train size:",len(train))
print("train Percentage of main dataset :",(len(train)/len(df))*100)
print("test size:",len(test))
print("test Percentage of dataset :",(len(test)/len(df))*100)
```

```
train size: 31887
train Percentage of main dataset : 79.9994982312652
test size: 7972
test Percentage of dataset : 20.00050176873479
```

2. If we had to, how would we prove to ourselves or a colleague that our data was indeed randomly sampled on X? And by prove, I mean empirically, not just showing this person our code. Don't actually do the work, just describe in your own words a test you could here. Hint: think about this in terms of selection bias and use notes from our 2nd lecture.

```
The selection bias can be present in the train and test datasets due to
small sample sizes or human errors such as typos,etc in the data set. Also
we know that, if, for a given i, being in sample is independent of Xi and
Yi values, then the sample is not biased, and we have a random sample. So
in our case we can build a model to predict Pr(train|X) with train and
test split datasets. A classifier can be build and target variable can be
predicted. We can prove our sample as random if there is no bias present.
Tests such as calculating accuracy of the model, if accuracy results
AUC>=0.5, we can prove our sample to be random.
```

3. Now build and train a decision tree classifier using `DecisionTreeClassifier()` (manual page) (http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html) on train_df to predict the `"churndep"` target variable. Make sure to use `criterion='entropy'` when instantiating an instance of `DecisionTreeClassifier()`. For all other settings you should use all of the default options.

In [33]:
```
from sklearn.tree import DecisionTreeClassifier

# Code here
clf = DecisionTreeClassifier(criterion='entropy')
clf.fit(train.drop('churndep',1), train['churndep'])
```

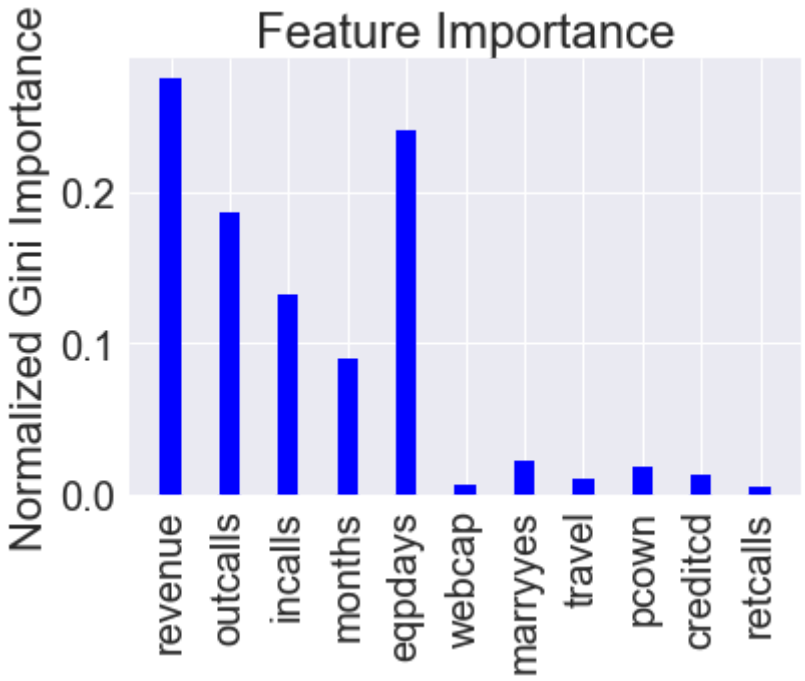
Out[33]:
```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=
None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=None, splitter='best')
```

4. Using the resulting model from 2.2, show a bar plot of feature names and their feature importance

(hint: check the attributes of the `DecisionTreeClassifier()` object directly in IPython or check the manual!).

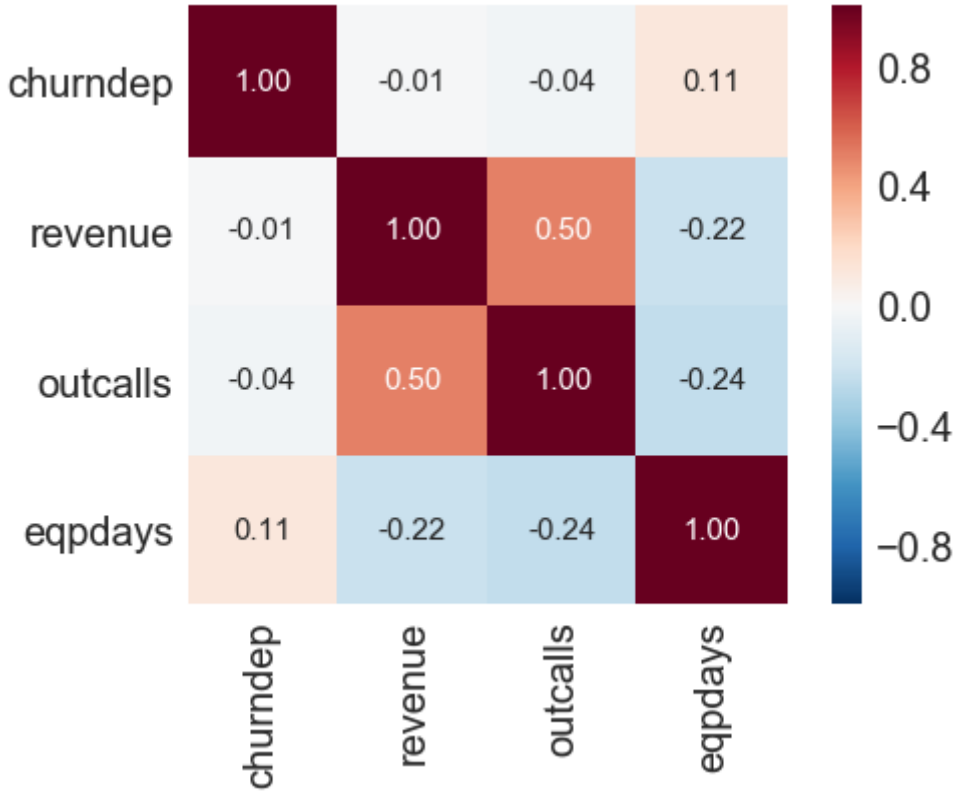
In [32]:
```
import matplotlib.pyplot as plt
%matplotlib inline

fig, ax = plt.subplots()
width = 0.35
ax.bar(np.arange(len(clf.feature_importances_)), clf.feature_importances_, w
ax.set_xticks(np.arange(len(clf.feature_importances_)))
ax.set_xticklabels(train.drop('churndep',1).columns.values, rotation = 90)
plt.title('Feature Importance')
ax.set_ylabel('Normalized Gini Importance')
```

Out[32]: `<matplotlib.text.Text at 0x11a97f9e8>`



5. Is the relationship between the top 3 most important features (as measured here) negative or positive? If your marketing director asked you to explain the top 3 drivers of churn, how would you interpret the relationship between these 3 features and the churn outcome? What "real-life" connection can you draw between each variable and churn?

In [30]:
```
# Code/answer here
import seaborn as sns
cols = ['churndep','revenue','outcalls','eqpdays']
cm = np.corrcoef(df[cols].values.T)
sns.set(font_scale=2)
hm = sns.heatmap(cm,
                 cbar=True,
                 annot=True,
                 square=True,
                 fmt='.2f',
                 annot_kws={'size': 15},
                 yticklabels=cols,
                 xticklabels=cols)
plt.show()
```



As per the correlation matrix and heatmap , the epqdays is positively correlated while revenue and outcalls are negatively correlated with churndep. Revenue and outcalls are postively correlated. As outcalls is mean number of outbound voice calls and revenue is mean monthly reveue in dollars, we can infer that customers with more phone usage are not likely to churn. Epqdays is negatively correlated to revenue and outcalls. It is positively correlatedwith churndep, hence we can infer customers using same equipment for longer period are more likely to churn as they may lookout for better offers.

6. Using the classifier built in 2.2, try predicting `"churndep"` on both the train_df and test_df data sets. What is the accuracy on each?

In [31]:
```
# Code here
from sklearn.metrics import accuracy_score,confusion_matrix
churndep_train = clf.predict(train.drop('churndep',1))
churndep_test = clf.predict(test.drop('churndep',1))
score_train = accuracy_score(train['churndep'],churndep_train)
score_test = accuracy_score(test['churndep'],churndep_test)
print("Predicted values of churndep for train: ",churndep_train)
print("Predicted values of churndep for test: ",churndep_test)
print("Accuracy_train : " ,score_train)
print("Accuracy_test : " ,score_test)
```

```
Predicted values of churndep for train:  [0 0 0 ..., 0 0 0]
Predicted values of churndep for test:  [0 1 1 ..., 0 0 0]
Accuracy_train :  0.999780474802
Accuracy_test :  0.534495735073
```

Part 2 - Finding a Good Decision Tree

The default options for your decision tree may not be optimal. We need to analyze whether tuning the parameters can improve the accuracy of the classifier. For the following options `min_samples_split` and `min_samples_leaf`:

1. Generate a list of 10 values of each for the parameters mim_samples_split and min_samples_leaf.

In [40]:
```
# Code here

min_samples_split_values = [100,200,400,500,700,1000,1200,1500,1700,2000]
min_samples_leaf_values = [1,2,5,20,50,100,150,200,350,500]
```

2. Explain in words your reasoning for choosing the above ranges.

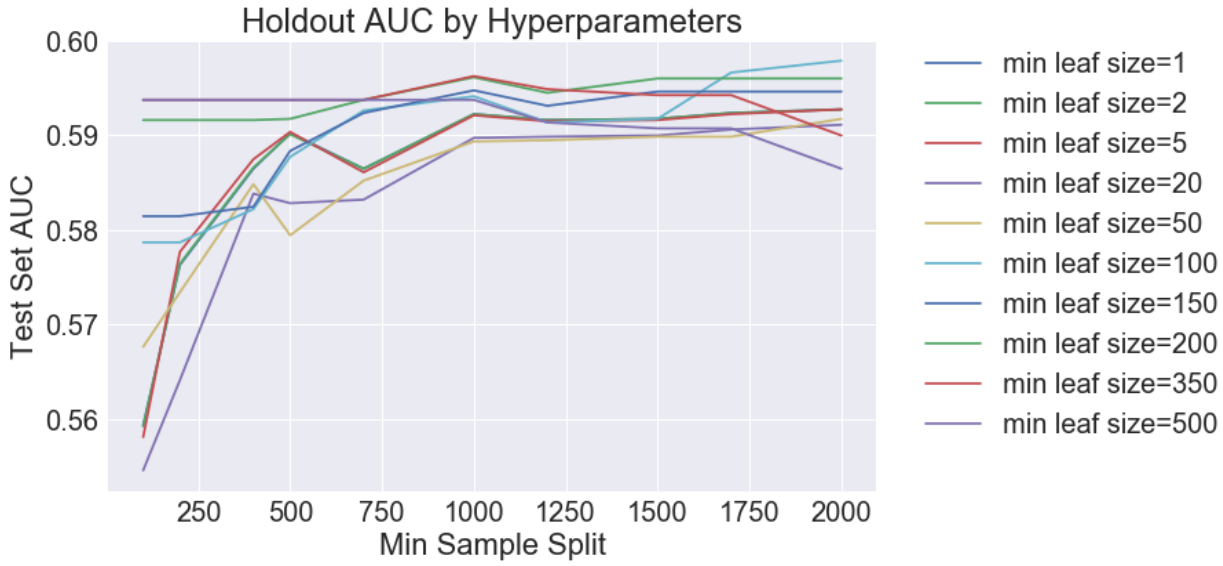
As there is no exact methodology to decide the range of values for the min_samples_split_values and min_samples_leaf_values, a wide variety of range is required. There is no prior knowledge of problem, varoius ranges should be tried such that no internal nodes that are "too small" are sensibly divided using min_split_values , also considering that each terminal has sufficient data using min_leaf_size.

3. For each combination of values in 3.1 (there should be 100), build a new classifier and check the classifier's accuracy on the test data. Plot the test set accuracy for these options. Use the values of `min_samples_split` as the x-axis and generate a new series (line) for each of `min_samples_leaf`.

In [41]:
```
import matplotlib.pyplot as plt
%matplotlib inline
accuracy_test = {}
accs1 = {}
# Code here
for i in min_samples_leaf_values:
    accuracy_test[i] = []
    accs1[i] = []
    for j in min_samples_split_values:
            clf = DecisionTreeClassifier(criterion = 'entropy', min_samples_
            clf = clf.fit(train.drop('churndep', 1), train['churndep'])
            churndep_test = clf.predict(test.drop('churndep',1))
            churndep_train = clf.predict(train.drop('churndep',1))
            accuracy_test[i].append(accuracy_score(test['churndep'],churndep
```

In [42]:
```
fig = plt.figure(figsize = (10, 6))
ax=fig.add_subplot(111)
for k in range(len(min_samples_leaf_values)):
    plt.plot(min_samples_split_values,accuracy_test[min_samples_leaf_values[k

plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0)
ax.set_xlabel('Min Sample Split')
ax.set_ylabel('Test Set AUC')
plt.title('Holdout AUC by Hyperparameters')
```

Out[42]: <matplotlib.text.Text at 0x11b77b588>



4. Which configuration returns the best accuracy? What is this accuracy? (Note, if you don't see much variation in the test set accuracy across values of min_samples_split or min_samples_leaf, try redoing the above steps with a different range of values).

In [43]:
```
# Code here
clf = DecisionTreeClassifier(criterion = 'entropy', min_samples_leaf = 100,
clf.fit(train.drop('churndep',1), train['churndep'])
ypred = clf.predict(test.drop('churndep',1))
confusion_matrix(test['churndep'], ypred), accuracy_score(test['churndep'],
```

Out[43]:
```
(array([[2026, 1975],
        [1231, 2740]]), 0.59784244856999502)
```

As per the Accuracy plot above and classifier trained, the best accuracy for the model trained is at min_samples_split = 2000 and min_samples_leaf_values =100. Thus it can be inferred that the mid range min_samples_leaf_values and high splits of min_samples_split draws good results. Overall the best accuracy ranges between 0.59 to 0.60 i.e 59%-60% . Also the best accuracy depends on the exact split and leaf sizes selected.

5. If you were working for a marketing department, how would you use your churn production model in a real business environment? Explain why churn prediction might be good for the business and how one might improve churn by using this model.

Churn model would prove to be important for the business environment, so the marketing team can design marketing strategies for the company to reduce customer churning. As we can infer that customer churn can have a significant impact on the revenue and reputation of the company, churn model can provide insights for the most significant factors affecting the customer churn or negative review. The churn model to provide better offers to the customers churning, has a positive effect on influential customers - the customers who churn on basis of other churning customers review. By using this model fast immediate effect strategies can be implemented which provide discounts and incentives to customers likely to churn. For example, offering new devices to the customers using same equipment for long time. Many strategis can be implemented to prevent customers churn in future. For example, the reducing outcall rates for non-frequent users, also providing different incentives on the equipments. The company be benefitted and can think of many ways to prevent customer churn on basis of churn model.

In [ ]:

In [ ]: