Decision Tree and Random Forest

The Decision Trees and Random Forests are two versatile machine learning models that are applicable to many machine learning tasks. In this lab, we will apply them to the classification problem

For this project we will be exploring publicly available data from LendingClub.com (www.lendingclub.com). Lending Club connects people who need money (borrowers) with people who have money (investors). Hopefully, as an investor you would want to invest in people who showed a profile of having a high probability of paying you back. We will try to create a model that will help predict this.

Lending club had a very interesting year in 2016 (https://en.wikipedia.org/wiki/Lending_Club#2016), so let's check out some of their data and keep the context in mind. This data is from before they even went public.

We will use lending data from 2007-2010 and be trying to classify and predict whether or not the borrower paid back their loan in full. You can download the data from here (https://www.lendingclub.com/info/download-data.action) or just use the csv already provided. It's recommended you use the csv provided as it has been cleaned of NA values.

Here are what the columns represent:

- credit.policy: 1 if the customer meets the credit underwriting criteria of LendingClub.com, and 0 otherwise.
- purpose: The purpose of the loan (takes values "credit_card", "debt_consolidation", "educational", "major_purchase", "small_business", and "all_other").
- int.rate: The interest rate of the loan, as a proportion (a rate of 11% would be stored as 0.11). Borrowers judged by LendingClub.com to be more risky are assigned higher interest rates.
- installment: The monthly installments owed by the borrower if the loan is funded.
- log.annual.inc: The natural log of the self-reported annual income of the borrower.
- dti: The debt-to-income ratio of the borrower (amount of debt divided by annual income).
- fico: The FICO credit score of the borrower.
- days.with.cr.line: The number of days the borrower has had a credit line.
- revol.bal: The borrower's revolving balance (amount unpaid at the end of the credit card billing cycle).
- revol.util: The borrower's revolving line utilization rate (the amount of the credit line used relative to total credit available).
- inq.last.6mths: The borrower's number of inquiries by creditors in the last 6 months.
- delinq.2yrs: The number of times the borrower had been 30+ days past due on a payment in the past 2 years.
- pub.rec: The borrower's number of derogatory public records (bankruptcy filings, tax liens, or judgments).

Import Libraries

Import the usual libraries for pandas and plotting. You can import sklearn later on.

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Get the Data

In [ ]:
```
loans = pd.read_csv('loan_data.csv')
```

In [ ]:
```
loans.info()
```

In [ ]:
```
loans.head()
```

In [ ]:
```
loans.describe()
```

Exploratory Data Analysis

a histogram of two FICO distributions on top of each other, one for each credit.policy outcome.

In [ ]:
```
loans[loans['credit.policy']==1]['fico'].hist(bins=30,figsize=(10,8),alpha=0.8,colo
r='orange',label='Credit Policy = 1')
loans[loans['credit.policy']==0]['fico'].hist(bins=30,figsize=(10,8),alpha=0.8,colo
r='G',label='Credit Policy = 0')
plt.xlabel('FICO')
plt.legend()
```

In [ ]:
```
loans[loans['not.fully.paid']==0]['fico'].hist(bins=30,figsize=(10,8),alpha=0.8,col
or='purple',label='Fully Paid')
loans[loans['not.fully.paid']==1]['fico'].hist(bins=30,figsize=(10,8),alpha=0.8,col
or='yellow',label='Not Fully Paid')
plt.legend()
plt.xlabel('FICO')
```

showing the counts of loans by purpose, with the color hue defined by not.fully.paid.

In [ ]:
```
fig, ax = plt.subplots(figsize=(10,8))
sns.countplot(x='purpose',data=loans,hue='not.fully.paid',palette='prism')
plt.tight_layout()
```

Let's see the trend between FICO score and interest rate.

In [ ]:
```
sns.jointplot(x='fico',y='int.rate',data=loans,color='purple')
```

lmplots to see the trend differed between not.fully.paid and credit.policy.

In [ ]:
```
sns.lmplot('fico','int.rate',data=loans,col='not.fully.paid',hue='credit.policy',pa
lette='gist_stern')
```

Setting up the Data

Let's get ready to set up our data for our Random Forest Classification Model!

In [ ]:
```
loans.info()
```

Categorical Features

Notice that the **purpose** column as categorical

That means we need to transform them using dummy variables so sklearn will be able to understand them. Let's do this in one clean step using pd.get_dummies.

In [ ]:
```
cat_feats = ['purpose']
```

In [ ]:
```
final_data = pd.get_dummies(loans,columns=cat_feats,drop_first=True)
```

Train Test Split

Now its time to split our data into a training set and a testing set!

In [ ]:
```
from sklearn.model_selection import train_test_split
```

In [ ]:
```
X = final_data.drop('not.fully.paid',axis=1)
y = final_data['not.fully.paid']
```

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_st
ate=42)
```

Training a Decision Tree Model

Let's start by training a single decision tree first!

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

In [ ]:
```
dtree = DecisionTreeClassifier()
```

In [ ]:
```
dtree.fit(X_train,y_train)
```

Predictions and Evaluation of Decision Tree

In [ ]:
```
pred = dtree.predict(X_test)
```

In [ ]:
```
from sklearn.metrics import confusion_matrix,classification_report
```

In [ ]:
```
print(classification_report(y_test,pred))
print(confusion_matrix(y_test,pred))
```

Training the Random Forest model

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [ ]:
```
rfc = RandomForestClassifier(n_estimators=100)
```

In [ ]:
```
rfc.fit(X_train,y_train)
```

Predictions and Evaluation

Let's predict off the y_test values and evaluate our model.

In [ ]:
```
rfc_pred = rfc.predict(X_test)
```

In [ ]:
```
print(classification_report(y_test,rfc_pred))
```

In [ ]:
```
print(confusion_matrix(y_test,rfc_pred))
```

What performed better the random forest or the decision tree?

Both the models,i.e. Random forest and decision tree, performed well. Although in depends on various factors, but we can approximately say that Random forest performance is better overall.

Exercises

*Give the Titanic data file with records found on Kaggle. Your task uses Random Forest to build a classifier which can predict whether they survived or not?

- Link: https://www.dropbox.com/request/YGBDBlBixkjtlfXGptLb

- Deadline: 10:00 am next Tuesday