Decision Trees and Random Forests

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Get the Data

```
loan = pd.read_csv('/content/loan_data.csv')
```

```
loan.head()
```

	credit.policy	purpose	int.rate	installment	log.annual.inc	dti	fico
0	1	debt_consolidation	0.1189	829.10	11.350407	19.48	737
1	1	credit_card	0.1071	228.22	11.082143	14.29	707
2	1	debt_consolidation	0.1357	366.86	10.373491	11.63	682
3	1	debt_consolidation	0.1008	162.34	11.350407	8.10	712
4	1	credit_card	0.1426	102.92	11.299732	14.97	667

Train Test Split

```
from sklearn.model_selection import train_test_split
```

```
feats=['purpose']
df=pd.get_dummies(loan,columns=feats,drop_first=True)
```

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

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)
```

Decision Trees

```
from sklearn.tree import DecisionTreeClassifier
```

```
dtree = DecisionTreeClassifier()
```

```
dtree.fit(X_train,y_train)
```

```
DecisionTreeClassifier()
```

Prediction and Evaluation

```
predictions = dtree.predict(X_test)
```

```
from sklearn.metrics import classification_report,confusion_matrix
```

```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.85      0.84      0.84      2391
           1       0.24      0.26      0.25       483

    accuracy                           0.74      2874
```

```
           macro avg       0.55      0.55      0.55      2874
        weighted avg       0.75      0.74      0.74      2874
```

```
print(confusion_matrix(y_test,predictions))
```

```
[[2001  390]
 [ 358  125]]
```

Tree Visualization

```
from IPython.display import Image
from six import StringIO
from sklearn.tree import export_graphviz
import pydot

features = list(df.columns[1:])
features
```

```
['int.rate',
 'installment',
 'log.annual.inc',
 'dti',
 'fico',
 'days.with.cr.line',
 'revol.bal',
 'revol.util',
 'inq.last.6mths',
 'delinq.2yrs',
 'pub.rec',
 'not.fully.paid',
 'purpose_credit_card',
 'purpose_debt_consolidation',
 'purpose_educational',
 'purpose_home_improvement',
 'purpose_major_purchase',
 'purpose_small_business']
```

```
dot_data = StringIO()
export_graphviz(dtree, out_file=dot_data,feature_names=features,filled=True,rounded=True)

graph = pydot.graph_from_dot_data(dot_data.getvalue())
Image(graph[0].create_png())
```

Random Forests

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=100)
rfc.fit(X_train, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
rfc_pred = rfc.predict(X_test)
```

```
print(confusion_matrix(y_test,rfc_pred))
```

```
[[2374   17]
 [ 478    5]]
```

```
print(classification_report(y_test,rfc_pred))
```

```
                   precision    recall  f1-score   support

               0       0.83      0.99      0.91      2391
               1       0.23      0.01      0.02       483
```

```
              accuracy                           0.83      2874
             macro avg       0.53      0.50      0.46      2874
          weighted avg       0.73      0.83      0.76      2874
```

Pronto!