AIN 180 Inclass Assignment Due: beginning of next class
(but complete outside of class)

Type the following in Notepad or Python as discussed in class.

Then print and turn in a hardcopy for credit **AND** upload the file to Canvas as well. Your grade for this inclass assignment will be based on how "error-free" it is, so be careful and attentitive.

```
# Now the algorithm becomes a Classification problem
# ************************************************
# Step 1:  Read in the data
import pandas as pd
import problem3 as p3

cars = pd.read_csv('carsmore.csv')
print()

p3.cardatasetsense(cars)
cars = p3.recode_brand(cars)

print('****  Recoded data ******')
print(cars)
print()
print('Number of data records and columns of data: ' , cars.shape)
print()

# ************************************************
# Step 2: Separate features from the target (result)
#
X = cars.iloc[:, :-1]
y = cars.iloc[:, [-1]]
print()
print('X is ')
print(X)
print()
print('y is ', y)
```

```
# ************************************************
# Step 3: Split into training and test sets
#
# Split at 70% and 30%
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test = train_test_split(X, y, test_size=0.3,random_state=0)
p3.print_splits(X_train, y_train,1)
p3.print_splits(X_test, y_test,2)


# ************************************************
# Step 4: Random Forest Classifier

from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train,y_train.values.ravel())

p3.print_feature(rf, X_train)


# ************************************************
# Step 5: Decision tree
# Then assess the test data (how it performs from the training)

from sklearn import tree
tree_model = tree.DecisionTreeClassifier()
tree_model = tree_model.fit(X_train, y_train)

# Now use test data for predictions
y_predicted = pd.DataFrame(tree_model.predict(X_test))
probs = pd.DataFrame(tree_model.predict_proba(X_test))
```

```
# ******************************************************
# Step 6 Check metrics
from sklearn import metrics
tree_accuracy = metrics.accuracy_score(y_test,y_predicted)
print()
print('*******************************')
print('Tree accuracy:')
print(tree_accuracy)
y_predicted.rename( columns={0 :'brand'}, inplace=True)
print('*************************************************************')
p3.display_predicted(y_test,y_predicted)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_predicted))
#read the diagonal which gives us the match between predicted (y)and expected (x)
print(classification_report(y_test, y_predicted))


# ******************************************************
# Step 7 Predict for a new input: (should be US (=1) and Europe(=3)

new_predict =tree_model.predict([[18,6,225,85,3465,17,1982]])
new_probs = tree_model.predict_proba([[18,6,225,85,3465,17,1982]])

p3.print_prediction(new_predict, new_probs)

new_predict2=tree_model.predict([[32,4,89,71,1925,14,1980]])
new_probs2 = tree_model.predict_proba([[32,4,89,71,1925,14,1980]])

p3.print_prediction(new_predict2, new_probs2)
```