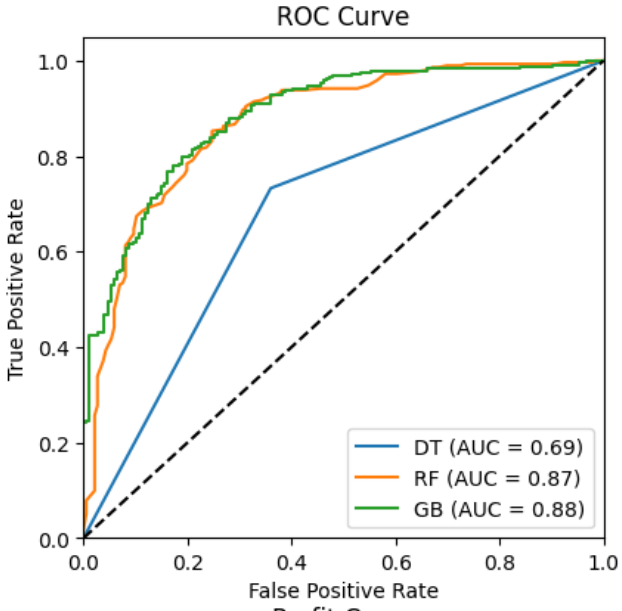
In [288…
```
import pandas as pd
import numpy as np
# import statsmodels.api as sm
# import statsmodels.formula.api as smf
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import roc_curve, auc, precision_recall_curve, confusion_matrix
import matplotlib.pyplot as plt
```

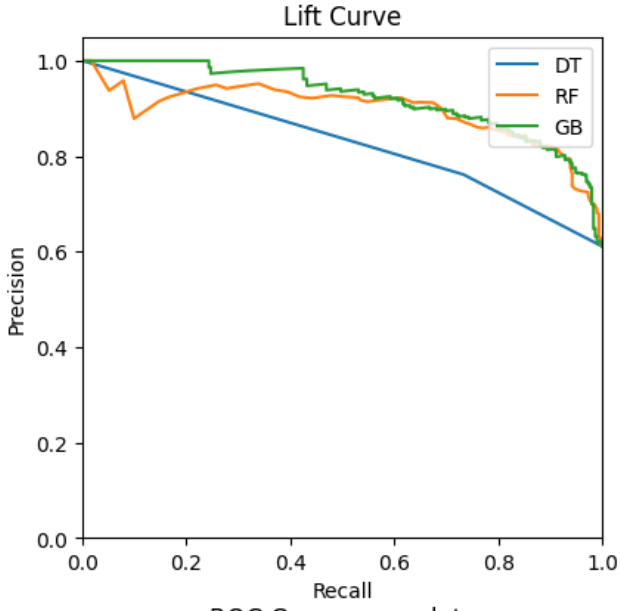
In [340…
```
fig, ax = plt.subplots(nrows=3, ncols=2, figsize=(10, 15))

df = pd.read_csv("~/Downloads/STCdata_A.csv",index_col=0)

le = LabelEncoder()
df = df.apply(le.fit_transform)

X = df.drop("Retained.in.2012.", axis=1)
y = df["Retained.in.2012."]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

dt = DecisionTreeClassifier(random_state=0)
dt.fit(X_train, y_train)

rf = RandomForestClassifier(random_state=0)
rf.fit(X_train, y_train)

gb = GradientBoostingClassifier(random_state=0)
gb.fit(X_train, y_train)

y_pred_dt = dt.predict_proba(X_test)
y_pred_rf = rf.predict_proba(X_test)
y_pred_gb = gb.predict_proba(X_test)

fpr_dt, tpr_dt, _ = roc_curve(y_test, y_pred_dt[:,1])
roc_auc_dt = auc(fpr_dt, tpr_dt)

fpr_rf, tpr_rf, _ = roc_curve(y_test, y_pred_rf[:,1])
roc_auc_rf = auc(fpr_rf, tpr_rf)

fpr_gb, tpr_gb, _ = roc_curve(y_test, y_pred_gb[:,1])
roc_auc_gb = auc(fpr_gb, tpr_gb)

plt.subplot(3, 2, 1)
plt.plot(fpr_dt, tpr_dt, label='DT (AUC = %0.2f)' % roc_auc_dt)
plt.plot(fpr_rf, tpr_rf, label='RF (AUC = %0.2f)' % roc_auc_rf)
plt.plot(fpr_gb, tpr_gb, label='GB (AUC = %0.2f)' % roc_auc_gb)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc="lower right")

precision, recall, thresholds = precision_recall_curve(y_test, y_pred_dt[:,1])
plt.subplot(3, 2, 2)
```

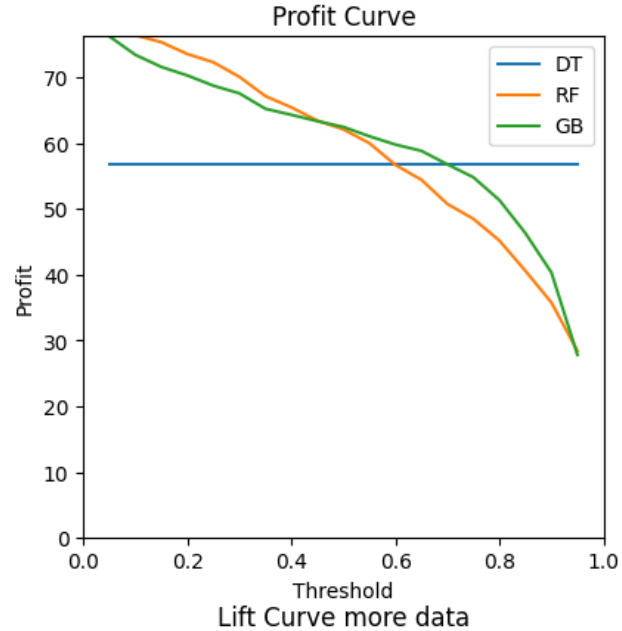
```
plt.plot(recall, precision, label='DT')

precision, recall, thresholds = precision_recall_curve(y_test, y_pred_rf[:,1])
plt.plot(recall, precision, label='RF')

precision, recall, thresholds = precision_recall_curve(y_test, y_pred_gb[:,1])
plt.plot(recall, precision, label='GB')

plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Lift Curve')
plt.legend(loc="upper right")

cost = 40
revenue = 100
thresholds = [0.05*i for i in range(1,20)]
profits = []
for threshold in thresholds:
    y_pred_dt_bin = y_pred_dt[:,1] >= threshold
    cm = confusion_matrix(y_test, y_pred_dt_bin)
    total_cost = cost*(cm[0,1] + cm[1,0]) + revenue*cm[1,1]
    profits.append(total_cost/len(y_test))

plt.subplot(3, 2, 3)
plt.plot(thresholds, profits, label='DT')

thresholds = [0.05*i for i in range(1,20)]
profits = []
for threshold in thresholds:
    y_pred_rf_bin = y_pred_rf[:,1] >= threshold
    cm = confusion_matrix(y_test, y_pred_rf_bin)
    total_cost = cost*(cm[0,1] + cm[1,0]) + revenue*cm[1,1]
    profits.append(total_cost/len(y_test))

plt.plot(thresholds, profits, label='RF')

thresholds = [0.05*i for i in range(1,20)]
profits = []
for threshold in thresholds:
    y_pred_gb_bin = y_pred_gb[:,1] >= threshold
    cm = confusion_matrix(y_test, y_pred_gb_bin)
    total_cost = cost*(cm[0,1] + cm[1,0]) + revenue*cm[1,1]
    profits.append(total_cost/len(y_test))

plt.plot(thresholds, profits, label='GB')

plt.xlim([0.0, 1.0])
plt.ylim([0.0, max(profits)])
plt.xlabel('Threshold')
plt.ylabel('Profit')
plt.title('Profit Curve')
plt.legend(loc="upper right")

# again with extra data
df_m = pd.read_csv("~/Downloads/STCdata_merged.csv",index_col=0)

le = LabelEncoder()
df_m = df_m.apply(le.fit_transform)

X = df_m.drop("Retained.in.2012.", axis=1)
y = df_m["Retained.in.2012."]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

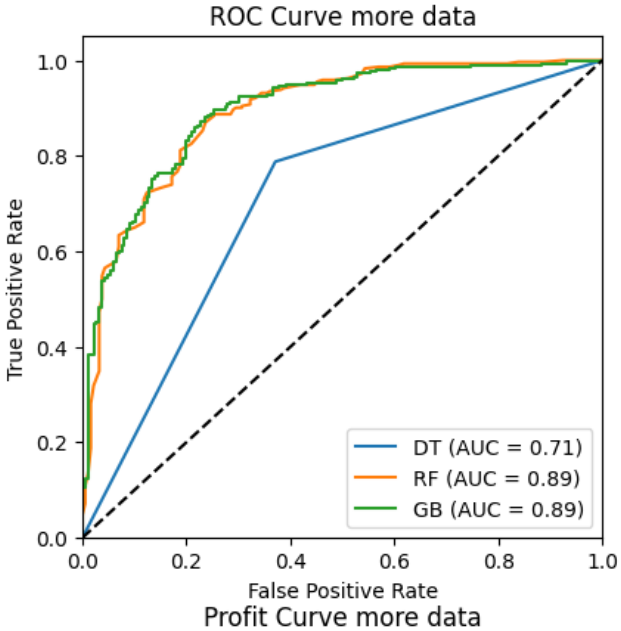
```
dt = DecisionTreeClassifier(random_state=0)
dt.fit(X_train, y_train)

rf = RandomForestClassifier(random_state=0)
rf.fit(X_train, y_train)

gb = GradientBoostingClassifier(random_state=0)
gb.fit(X_train, y_train)

y_pred_dt = dt.predict_proba(X_test)
y_pred_rf = rf.predict_proba(X_test)
y_pred_gb = gb.predict_proba(X_test)

fpr_dt, tpr_dt, _ = roc_curve(y_test, y_pred_dt[:,1])
roc_auc_dt = auc(fpr_dt, tpr_dt)

fpr_rf, tpr_rf, _ = roc_curve(y_test, y_pred_rf[:,1])
roc_auc_rf = auc(fpr_rf, tpr_rf)

fpr_gb, tpr_gb, _ = roc_curve(y_test, y_pred_gb[:,1])
roc_auc_gb = auc(fpr_gb, tpr_gb)

plt.subplot(3, 2, 4)
plt.plot(fpr_dt, tpr_dt, label='DT (AUC = %0.2f)' % roc_auc_dt)
plt.plot(fpr_rf, tpr_rf, label='RF (AUC = %0.2f)' % roc_auc_rf)
plt.plot(fpr_gb, tpr_gb, label='GB (AUC = %0.2f)' % roc_auc_gb)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve more data')
plt.legend(loc="lower right")

precision, recall, thresholds = precision_recall_curve(y_test, y_pred_dt[:,1])
plt.subplot(3, 2, 5)
plt.plot(recall, precision, label='DT')

precision, recall, thresholds = precision_recall_curve(y_test, y_pred_rf[:,1])
plt.plot(recall, precision, label='RF')

precision, recall, thresholds = precision_recall_curve(y_test, y_pred_gb[:,1])
plt.plot(recall, precision, label='GB')

plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Lift Curve more data')
plt.legend(loc="upper right")

cost = 40
revenue = 100
thresholds = [0.05*i for i in range(1,20)]
profits = []
for threshold in thresholds:
    y_pred_dt_bin = y_pred_dt[:,1] >= threshold
    cm = confusion_matrix(y_test, y_pred_dt_bin)
    total_cost = cost*(cm[0,1] + cm[1,0]) + revenue*cm[1,1]
    profits.append(total_cost/len(y_test))
plt.subplot(3, 2, 6)
plt.plot(thresholds, profits, label='DT')
```

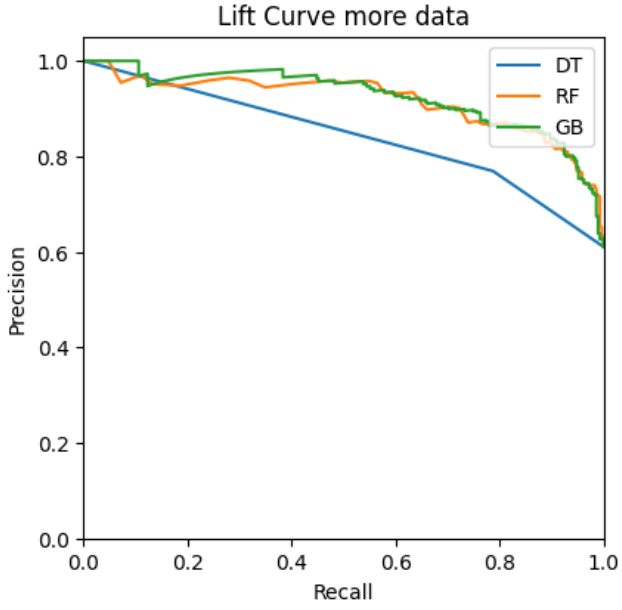
```
thresholds = [0.05*i for i in range(1,20)]
profits = []
for threshold in thresholds:
    y_pred_rf_bin = y_pred_rf[:,1] >= threshold
    cm = confusion_matrix(y_test, y_pred_rf_bin)
    total_cost = cost*(cm[0,1] + cm[1,0]) + revenue*cm[1,1]
    profits.append(total_cost/len(y_test))

plt.plot(thresholds, profits, label='RF')

thresholds = [0.05*i for i in range(1,20)]
profits = []
for threshold in thresholds:
    y_pred_gb_bin = y_pred_gb[:,1] >= threshold
    cm = confusion_matrix(y_test, y_pred_gb_bin)
    total_cost = cost*(cm[0,1] + cm[1,0]) + revenue*cm[1,1]
    profits.append(total_cost/len(y_test))

plt.plot(thresholds, profits, label='GB')

plt.xlim([0.0, 1.0])
plt.ylim([0.0, max(profits)])
plt.xlabel('Threshold')
plt.ylabel('Profit')
plt.title('Profit Curve more data')
plt.legend(loc="upper right")
plt.show()
```

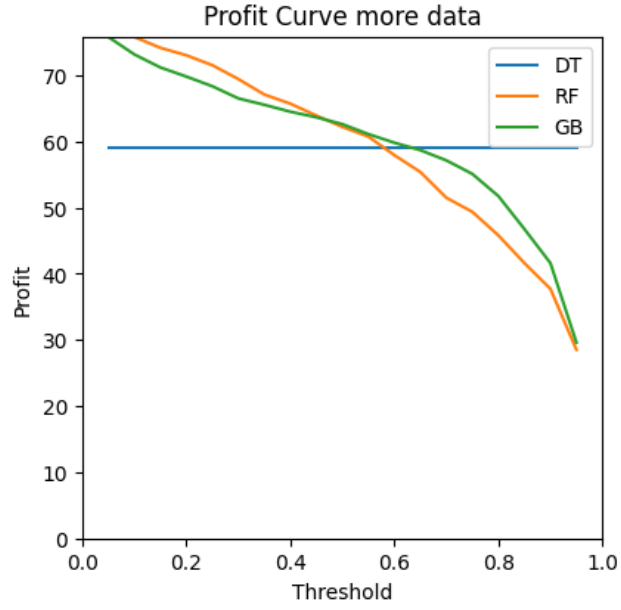












How do these methods compare to the logistic regression models?

The logistic model is underperforming these models (gradient boost and random forest). We can see this especially from the ROC curve. These models, especially gradient boosting and random forest, are doing better as seen by their AUC (0.87 for RF and 0.88 for GB, while 0.85 for logistic regression) where we know that a higher AUC is better.

Comment on the improvement (or lack thereof) from incorporating the NPS data

Including the data from Emily was a net improvement to our performance. For example, AUC went from 0.69,0.87,0.88 to 0.71,0.89,0.89 for decision tree, random forest and gradient boost respectively. We can see some to no improvement in lift curve and profit curve. In sum, while the improvement from getting more data is not large, it is still an improvement.

In [ ]: