THE UNIVERSITY OF CHICAGO
HARRIS SCHOOL OF PUBLIC POLICY

Machine Learning - Mini-Project (Excerpt)
Government Tax Revenue

PPHA 30545 - Professor Clapp

Student: Jeremy Xu

Motivation

The capacity of the government to collect taxes is pivotal to long-run economic growth because without tax revenue, the state cannot provide public goods. One way the government can increase tax revenue is by increasing the tax rate. Another way to increase tax revenue is by reducing the probability of successful tax evasion; as probability of success decreases, the incentive to cheat gets weaker. The government can reduce the probability of successful tax evasion by simply increasing the number of audits it performs. However, increasing the number of audits also increases government expenditure, which may offset the increase in tax revenue. Another way to reduce the probability of successful tax evasion is to better target audits. That is, the government can increase the probability of catching tax evasion by reducing the number of audits performed on firms that paid their taxes and increasing the number of audits performed on firms that evaded their taxes. How might the government go about such an effort?

This is a prediction problem, so the government can approach this effort using the machine learning techniques we're covering in class! For instance, one way the government can become more adept at going after firms that were dishonest on their taxes is by using a Linear Probability Model (LPM) or KNN (k-Nearest Neighbors). Firms that evade taxes might display similar characteristics, allowing the government to predict whether a firm has evaded taxes with a low classification error rate.

The dataset for this project contains information on firms that the government of India suspected of tax evasion and subsequently the Comptroller and Auditor General (CAG) of India performed audits on. Table 1 contains the variable names and their definitions. Naturally, the outcome variable is whether the auditor found that the firm evaded taxes as a result of the audit (Risk). The predictors include various quantitative measures about the firms.

Table 1: Variable Names and Definitions

Variable	Definition
`Risk`	1 if firm found to have evaded taxes after audit; 0 otherwise
`Sector`	Historical risk score for the industry sector of a firm
`PARA A`	Discrepancy found in planned expenditure (in crore)
`Risk A`	Risk score computed from Para A and firm traits
`PARA B`	Discrepancy found in unplanned expenditure (in crore)
`Risk B`	Risk score computed from Para B and firm traits
`Money Value`	Firm revenue in past 2 years
`Risk D`	Risk score computed from some firm traits
`Score`	Comprehensive risk score
`Inherent Risk`	Firm's historical risk score
`Audit Risk`	Total discrepancy score computed from examining firm tax returns

Note: 1 crore = 10 million rupees = 130,000 USD.

Forest-for-the-Trees Questions

1. Since it's important to use theory/intuition/common sense in concert with our data driven approaches, what factors do you suspect will affect the true, underlying model of whether or not a firm will commit tax evasion? Briefly explain.

   (1) The sample size of the data is not large enough, which may result in lower test prediction accuracy of the result.

   (2) Predictors such as Money_Value may have low correlation with the Risk outcome.

   (3) Predictors such as PARA_B and RISK_B may have some outliers in the dataset, where some values are particularly large compared to the mean.

   These are the potential factors that I suspect will affect the true, underlying model of whether or not a firm will commit tax evasion.

2. Assume that in addition to some combination of the predictors listed in Table 1, the interaction of two independent variables also enters the true model. Without explicitly having the interaction term as a predictor in the fitted model, what advantage does KNN enjoy over the LPM if the interaction variable is indeed important to the true relationship?

   Without explicitly having the interaction term as a predictor in the fitted model, KNN, as a non-parametric method for classification and regression, has the advantage of implicitly capturing the interaction effects between predictors.

Data Analysis Questions

3. Use the first half of the data (the first 388 units of observations henceforth) to train your linear probability model (LPM). Apply the model to the second half of the data to predict the probability a firm cheated.

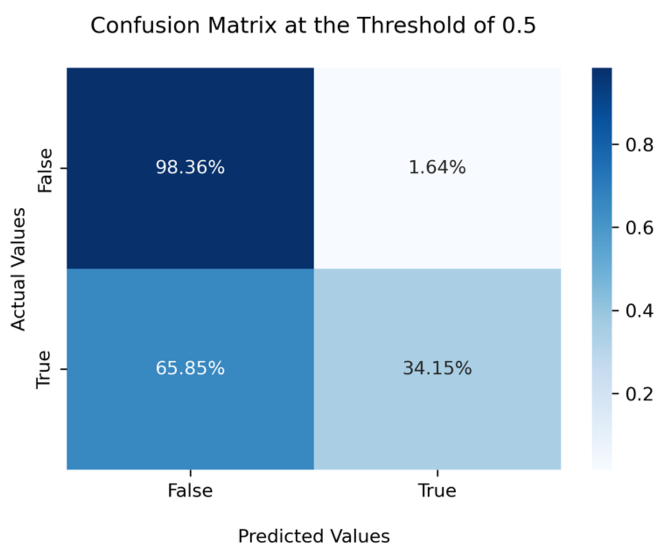
   (a) For firms with a predicted probability of tax evasion greater than 0.5, what proportion of the firms evaded taxes?

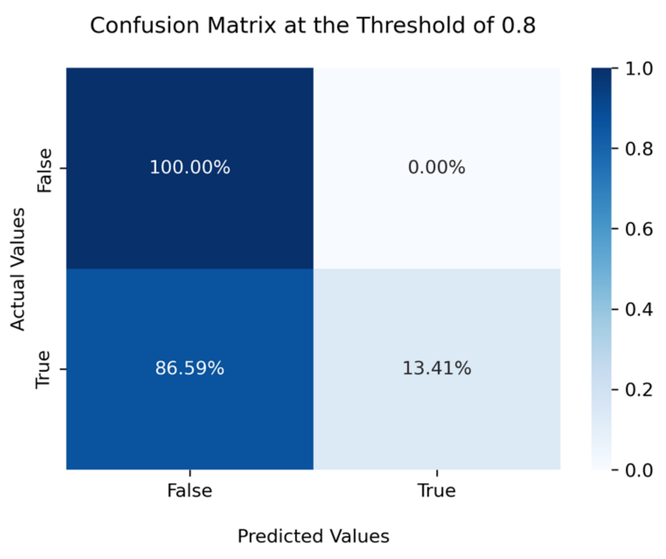
   For firms with a predicted probability of tax evasion greater than 0.5, $\frac{28}{28+5} = \mathbf{84.85}\%$ of the firms evaded taxes.

   (b) For firms with a predicted probability of tax evasion greater than 0.8, what fraction of the firms evaded taxes?

   For firms with a predicted probability of tax evasion greater than 0.8, **100%** of the firms evaded taxes.

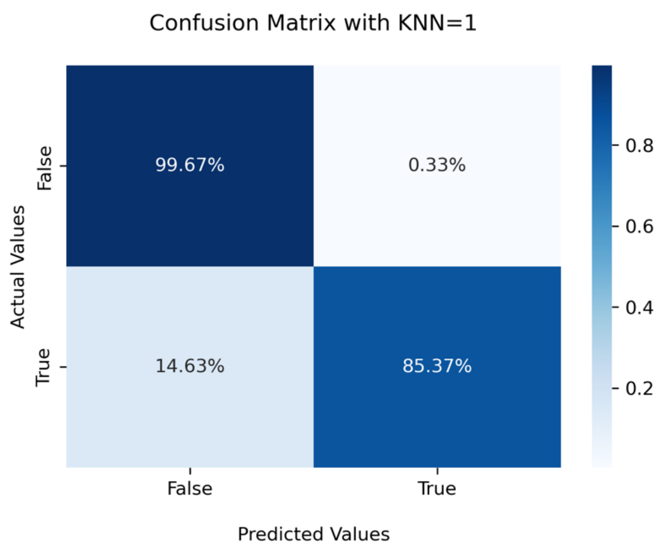
   (c) Construct the confusion matrices for both.





4. In measuring performance, should a false negative rate matter as much as a false positive rate? Briefly explain why or why not and how changing the threshold for classifying a firm as a tax evader (as in the previous question) affects this trade-off.
   In measuring performance, a false negative rate should matter more than a false positive rate. Since we would like to capture firms that would evade their taxes, a lower false negative rate is often more desirable to include as many tax evaders as possible. Increasing the threshold will increase the false negative rate, where a higher number of tax evaded firms will be omitted, and thus it would be a good trade-off to keep the threshold at a low level.

5. Using the first half of the data as training data, fit a KNN model with k = 1, then use it to predict outcomes in the testing data.
   (a) Construct the confusion matrix.



(b) With firms that are predicted for tax evasion, what fraction actually evaded taxes?

With firms that are predicted for tax evasion, $\frac{70}{70+1} = \mathbf{98.59}\%$ actually evaded taxes.

(c) What fraction of the firms that evaded taxes were predicted to have evaded taxes?

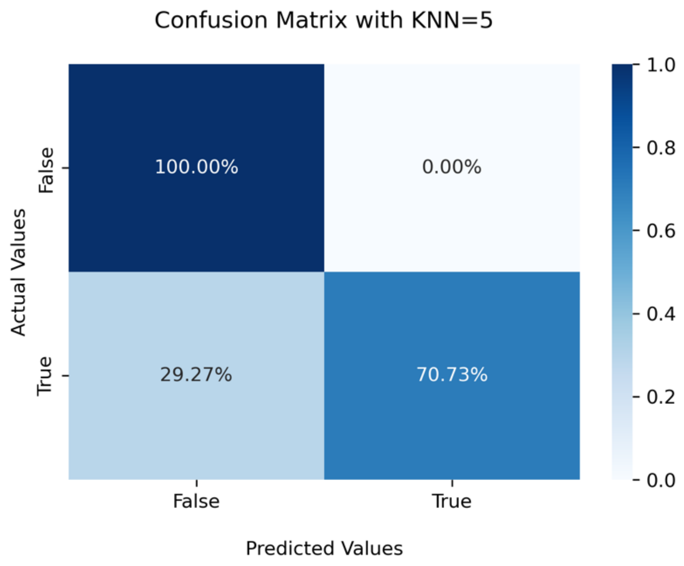
85.37% of the firms that evaded taxes were predicted to have evaded taxes (true positive).

(d) Determine whether KNN performs better with or without the attributes normalized.

The accuracy of KNN without the attributes normalized is **96.64%**, while the accuracy of KNN with the attributes normalized is **93.28%**. Therefore, KNN performs better without the attributes normalized.

6. Repeat the previous question with k = 5.

(a) Construct the confusion matrix.



(b) With firms that are predicted for tax evasion, what fraction actually evaded taxes?

With firms that are predicted for tax evasion, **100%** actually evaded taxes.

(c) What fraction of the firms that evaded taxes were predicted to have evaded taxes?

70.73% of the firms that evaded taxes were predicted to have evaded taxes (true positive).

(d) Determine whether KNN performs better with or without the attributes normalized.

The accuracy of KNN without the attributes normalized is **93.80%**, while the accuracy of KNN with the attributes normalized is **89.15%**. Therefore, KNN performs better without the attributes normalized.

7. Which KNN model performs better? Briefly explain how you make this determination and why you think this is the case.
   KNN model with k=1 performs better, as it has a higher accuracy than the KNN model with k=5. KNN model with k=1 has a higher flexibility than k=5, which results in lower bias and thus produces a more accurate prediction.

8. For KNN, which k yields the lowest error rate? By 5-fold cross-validation (5FCV), find the k with the lowest classification error rate. (Use the entire dataset for 5FCV, shuffle the data randomly for splitting, and set random_state = 13.)
   For KNN, k=1 yields the lowest error rate.
   By 5-fold cross-validation (5FCV), k=1 has the lowest classification error rate (97.94%).

9. Compare the optimal KNN from Problem 8 with the LPMs from Problem 3. Which is better? Follow these steps to answer. First, find how many firms are predicted by the optimal KNN to have committed tax fraud. Second, find the threshold q where you would "predict" the same number of firms as having committed tax fraud using the LPM. Then within the pool of firms that are classified as having evaded taxes, identify which has a lower proportion of firms that did not commit tax evasion.
   There are **71** firms predicted by the optimal KNN with k=1 to have committed tax fraud.
   I would "predict" the same number of firms as having committed tax fraud using the LPM at the threshold q of **0.27**.
   For optimal KNN model, the proportion of firms that did not commit tax evasion is $\frac{1}{71} = \mathbf{1.41}\%$; while for LPM model, the proportion is $\frac{24}{71} = \mathbf{33.80}\%$. Therefore, optimal KNN has a lower proportion of firms that did not commit tax evasion within the pool of firms that are classified as having evaded taxes, and thus is better.

10. In the long run, what problem might arise from the nature of the sample if the government heavily uses your best KNN model to target audits?
    When the KNN model is trained on data with only firms that were audited, it will result in a biased sample. The model may thus have poor prediction on firms that haven't been audited before and cause inaccuracy in decision making.

```python
### Import the libraries
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.formula.api as smf
from sklearn import preprocessing
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV, KFold

import warnings
from sklearn.exceptions import DataConversionWarning
warnings.filterwarnings(action='ignore', category=DataConversionWarning)
warnings.simplefilter("ignore", UserWarning)

### define path
path = r'/Users/jeremyxu/Desktop/2023 Winter/[Academic] PPHA 30546 Machine Learning/MP 02'
os.chdir(path)


### Preparing the Data
data = pd.read_csv('Data-Audit.csv')  # load dataset
data = data.dropna()  # drop NA data
describe = data.describe()
corr = data.corr()['Risk']  # correlation coefficients
print(corr)


### Data Analysis Questions
## split data
```

```
X_train = data.iloc[:388,:].drop(['Risk'], axis=1)
X_test = data.iloc[388:,:].drop(['Risk'], axis=1)
y_train = data.iloc[:388,10]
y_test = data.iloc[388:,10]


## (3) linear probability model
model = LinearRegression(fit_intercept=True)
model.fit(X_train, y_train)
y_pred_ols = model.predict(X_test)  # predict on test set

## (3a) tax evasion greater than 0.5
y_pred_ols_bin_1 = np.where(y_pred_ols > 0.5, 1, 0)
proportion_1 = np.count_nonzero(y_pred_ols_bin_1 == 1) / len(y_pred_ols_bin_1)
print(f'{round(proportion_1, 4)*100}%')

# confusion matrix
cm_ols_1 = confusion_matrix(y_test, y_pred_ols_bin_1, normalize = 'true')
print(confusion_matrix(y_test, y_pred_ols_bin_1))
plt.figure(figsize=(6,4))
ax = sns.heatmap(cm_ols_1, annot=True, fmt='.2%', cmap='Blues')
ax.set_title('Confusion Matrix at the Threshold of 0.5\n')
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values')
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])
plt.savefig('confusion_matrix_1.png', bbox_inches='tight')

## (3b) tax evasion greater than 0.8
y_pred_ols_bin_2 = np.where(y_pred_ols > 0.8, 1, 0)
proportion_2 = np.count_nonzero(y_pred_ols_bin_2 == 1) / len(y_pred_ols_bin_2)
print(f'{round(proportion_2, 4)*100}%')
```

```
# confusion matrix
cm_ols_2 = confusion_matrix(y_test, y_pred_ols_bin_2, normalize = 'true')
print(confusion_matrix(y_test, y_pred_ols_bin_2))
plt.figure(figsize=(6,4))
ax = sns.heatmap(cm_ols_2, annot=True, fmt='.2%', cmap='Blues')
ax.set_title('Confusion Matrix at the Threshold of 0.8\n')
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values')
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])
plt.savefig('confusion_matrix_2.png', bbox_inches='tight')


## (5) KNN model k=1
# KNN without normalization
knn_1 = KNeighborsClassifier(n_neighbors = 1)
knn_1.fit(X_train, y_train)
y_pred_knn1 = knn_1.predict(X_test)
print(accuracy_score(y_test, y_pred_knn1))

## (5a) confusion matrix KNN=1
cm_knn1 = confusion_matrix(y_test, y_pred_knn1, normalize = 'true')
print(confusion_matrix(y_test, y_pred_knn1))
plt.figure(figsize=(6,4))
ax = sns.heatmap(cm_knn1, annot=True, fmt='.2%', cmap='Blues')
ax.set_title('Confusion Matrix with KNN=1\n')
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values')
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])
plt.savefig('confusion_matrix_3.png', bbox_inches='tight')

## (5d) K=1 with normalization
```

```
X = data.drop(['Risk'], axis = 1)
y = data[['Risk']]
X_norm = pd.DataFrame(preprocessing.scale(X))
X_norm_train = X_norm.iloc[:388,:]
X_norm_test = X_norm.iloc[388:,:]
knn_norm_1 = KNeighborsClassifier(n_neighbors = 1)
knn_norm_1.fit(X_norm_train, y_train)
y_pred_norm_knn1 = knn_norm_1.predict(X_norm_test)
print(accuracy_score(y_test, y_pred_norm_knn1)) # with normalization
print(accuracy_score(y_test, y_pred_knn1)) # without normalization


## (6) KNN model k=5
# KNN without normalization
knn_5 = KNeighborsClassifier(n_neighbors = 5)
knn_5.fit(X_train, y_train)
y_pred_knn5 = knn_5.predict(X_test)
print(accuracy_score(y_test, y_pred_knn5))

## (6a) confusion matrix KNN=5
cm_knn5 = confusion_matrix(y_test, y_pred_knn5, normalize = 'true')
print(confusion_matrix(y_test, y_pred_knn5))
plt.figure(figsize=(6,4))
ax = sns.heatmap(cm_knn5, annot=True, fmt='.2%', cmap='Blues')
ax.set_title('Confusion Matrix with KNN=5\n')
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values')
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])
plt.savefig('confusion_matrix_5.png', bbox_inches='tight')

## (6d) K=5 with normalization
knn_norm_5 = KNeighborsClassifier(n_neighbors = 5)
```

```
knn_norm_5.fit(X_norm_train, y_train)
y_pred_norm_knn5 = knn_norm_5.predict(X_norm_test)
print(accuracy_score(y_test, y_pred_norm_knn5))  # with normalization
print(accuracy_score(y_test, y_pred_knn5))  # without normalization


## (8a) k yields the lowest error rate
def odd(n):
    return list(range(1, 2*n, 2))
ks = odd(100)
ac_rate = []
for i in ks:
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    ac_rate.append(np.mean(pred_i == y_test))

max_value = max(ac_rate)
opt_k = ac_rate.index(max_value)
print(opt_k)

## (8b) 5FCV
knni = KNeighborsClassifier()
para = {'n_neighbors':ks}
knn_cv = GridSearchCV(knni, para, cv = KFold(5, random_state=13, shuffle=True))
knn_cv.fit(X, y)
print(knn_cv.best_params_)
print(knn_cv.best_score_)


## (9) threshold q
y_pred_ols_bin_3 = np.where(y_pred_ols > 0.27, 1, 0)
print(confusion_matrix(y_test, y_pred_ols_bin_3)) # LPM threshold of 0.27
print(confusion_matrix(y_test, y_pred_knn1)) # KNN k=1
```