Question 1

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 ?

Following factors(but not limited to just these) I suspect would affect the true underlying model:

Risk : The risk score of the firm can provide valuable information on the firm's likelihood of committing tax evasion. It can be averaged across the industry for firms of similar standings and then can be applied to the firm in question to understand the likelihood of tax evasion.

Industry/Sector : The industry/sector also plays a role in possibility of a firm committing tax evasion. This is because of the underlying regulations and avenues to shied taxes.

Ownership or size of the firm: Smaller firms or the firms with individual owners have minimal resources and are possibly less likely to engage in tax evasion. Thus the ownership structure and the size of the firm is a relevant factor.

Political preferences: As observed in many countries, both developed and developing, firms with political inclination and preferences to the government usually have opportunities to evade taxes. This could be either by hiding taxes or by favourable tax policies and laws or due to no scrutiny by the government.

General accounting/auditing practices: The quality of a firm's accounting/auditing practices can also affect a firm's possibility of committing tax evasion.

Competition in industry: Firms in competitive industries may have more possibility to evade taxes. THis can be done to gain an advantage over the competitor firms in the industry.

Question 2

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?

K-Nearest Neighbors (KNN) and Linear Probability Model (LPM) are two different methods used in predictive modeling. KNN employs a non-parametric approach that ideally does not make any assumptions about the relationship between the variables provided and the outcome being predicted. However, KNN also looks at the characteristics of the closest neighbors to what is being predicted and uses that as a reference point for information to make the prediction. Such a methodoly allows the KNN model to capture more complex relationships between the variables and any non-linear patterns, even if they don ot follow a straight line relation.

Coming to the LPM model, it follows what is called a parametric approach. It is based on the assumption that the relationship between the variables and the outcome is linear, while also actively ignoring any non linear relationships or interactions between the variables. This is also a reason behind why sometimes if the relatiosnhip between the variables and the outcome is complex or contains a non-linear relationsip, LPM may not accurately determine the true relationship that exisits.

In similar situations, KNN would be deemed to be a better model as it can take care of non linear relationships including the interaction terms. This provides an advantage that KNN enjoys over the LPM if the interaction variable is indeed important to the true relationship.

Question 3

3. (10 points) Use the first half of the data (the first 388 units of observations henceforth) to train your linear probability model (LPM).5 Apply the model to the second half of the data to predict the probability a firm cheated.

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

In [1]:
```
import numpy as np
import pandas as pd
from sklearn import preprocessing
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import pyplot
from sklearn.model_selection import GridSearchCV, KFold
from sklearn.linear_model import LinearRegression
data = pd.read_csv('Data-Audit.csv').dropna()

data.head()
```

Out[1]:

	Sector_score	PARA_A	Risk_A	PARA_B	Risk_B	Money_Value	Risk_D	Score	Inherent_Risk	Audi
0	3.89	4.18	2.508	2.50	0.500	3.38	0.676	2.4	8.574	1
1	3.89	0.00	0.000	4.83	0.966	0.94	0.188	2.0	2.554	0
2	3.89	0.51	0.102	0.23	0.046	0.00	0.000	2.0	1.548	0
3	3.89	0.00	0.000	10.80	6.480	11.75	7.050	4.4	17.530	3
4	3.89	0.00	0.000	0.08	0.016	0.00	0.000	2.0	1.416	0

In [2]:
```
x = data.drop(['Risk'], axis = 1)

y = data['Risk']
```

In [3]:
```
print(x.shape)
print(y.shape)
```

```
(775, 10)
(775,)
```

In [4]:
```
x_train = x.head(388)
y_train = y.head(388)

x_test = x.tail(387)
y_test = y.tail(387)
```

In [5]:
```
lpm = LinearRegression()
lpm.fit(x_train, y_train)

y_pred_prob = lpm.predict(x_test)

y_pred = np.where(y_pred_prob > 0.5, 1, 0)

accuracy = (y_pred == y_test).mean()

print('Accuracy:', accuracy.round(2))


proportion = sum((y_pred_prob > 0.5) & (y_test == 1))/sum(y_pred_prob > 0.5)


print(proportion)
```

```
Accuracy: 0.85
0.8484848484848485
```

For the firms with a predicted probability of tax evasion greater than 0.5,the proportion of the firms that evade taxes is equal to approximately 85%

3b) For firms with a predicted probability of tax evasion greater than 0.8, what fraction of the firms evaded taxes?

In [6]:
```
#3b

lpm = LinearRegression()
lpm.fit(x_train, y_train)

y_pred_prob = lpm.predict(x_test)

y_pred = np.where(y_pred_prob > 0.8, 1, 0)

accuracy = (y_pred == y_test).mean()

print('Accuracy:', accuracy.round(2))

#Another method to find proportion
proportion = sum((y_pred_prob > 0.8) & (y_test == 1))/sum(y_pred_prob > 0.8)

print(proportion)
```

```
Accuracy: 0.82
1.0
```

For the firms with a predicted probability of tax evasion greater than 0.8,the proportion of the firms that evade taxes is equal to approximately 100%

3c) Construct the confusion matrices for both.

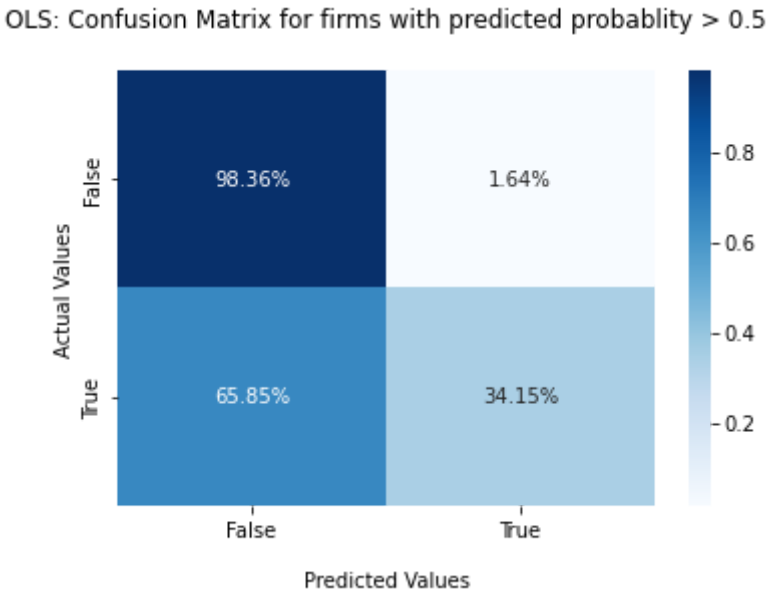
In [7]:
```
cm_lpm = confusion_matrix(y_test, y_pred_prob > 0.5, normalize = 'true')
ax = sns.heatmap(cm_lpm, annot=True,
```

```
            fmt='.2%', cmap='Blues')

ax.set_title('OLS: Confusion Matrix for firms with predicted probablity > 0.5\n')
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values')

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])

## Display the visualization of the Confusion Matrix.
plt.show()
```



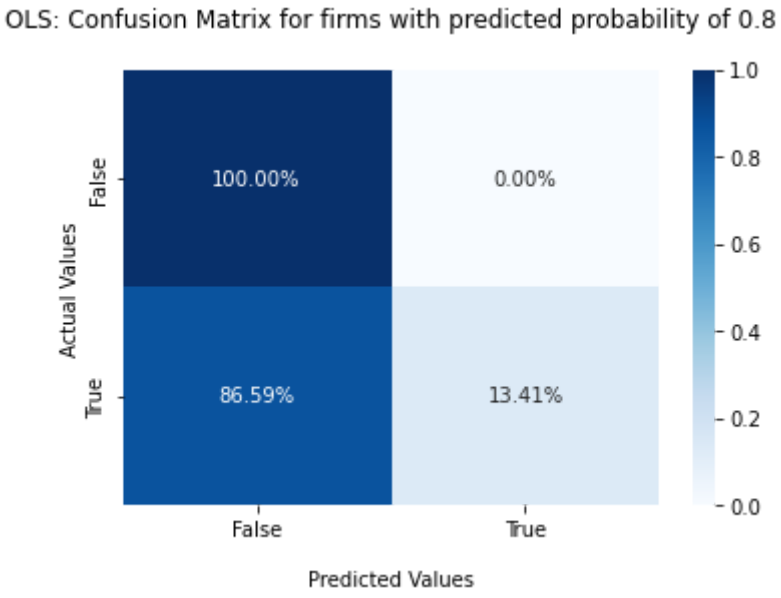
In [8]:

```
cm_lpm = confusion_matrix(y_test, y_pred_prob > 0.8, normalize = 'true')
ax = sns.heatmap(cm_lpm, annot=True,
            fmt='.2%', cmap='Blues')

ax.set_title('OLS: Confusion Matrix for firms with predicted probability of 0.8\n')
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values')

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])

## Display the visualization of the Confusion Matrix.
plt.show()
```



Question 4

4. (10 points) 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.

The question of whether one of the false positive rate and false negative rate is an important measure over the other depends on the context and the outcomes of such an error.

A situation is called as false positive, when the model predicts that a firm has evaded taxes, but actually the firm has not evaded taxes. Such an error would cause the administration to investigate unnecessarily and increases the costs associated with audits.

A situation is called false negative, when the model predicts that a firm has not evaded taxes, however actually the firm has evaded taxes. Such an error would allow the administration to miss out on the firms that are evading taxes and would result in a loss of taxes to the government.

Now to compare both the measures, we need to understand the cost associated with both the situations. If the cumulative cost of auditing and going after a firm is higher than the tax losses, the False positive situations needs to be reduced, even if that means that we are letting go of more firms that are classified in false negative group. In the other way, if the tax losses are higher than the cumulative cost of auditing and going after the firms, then the Flase negative occurances needs to be reduced, even though that means that wrongly accusing and investigating the firms that are not evading taxes.

Coming to the treshold i.e the probability at which the firms as classified as tax evaders or not directly affects the number of false positive and negative cases. If we assume a higher threshold, it results in a reduction in false positve cases, but the tradeoff would be reduced capability of identifying the firms that are tax evaders. On the other hand, if we assume a lower treshold, we would be more capable in identifying the firms that are tax evaders, but at the trade off of accusing more firms as tax evaders even when in reality they didnot. Thus an appropriate treshold needs to be determined by the government by considering the tradeoffs in both the situations before implementing the system.

Question 5

5. (10 points) 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.

Non Normalized

5(a) Construct the confusion matrix.

In [9]:
```
#No normalization

knn_1 = KNeighborsClassifier(n_neighbors = 1)
knn_1.fit(x_train, y_train)
y_pred_knn1 = knn_1.predict(x_test)
accuracy_score(y_test, y_pred_knn1)
```

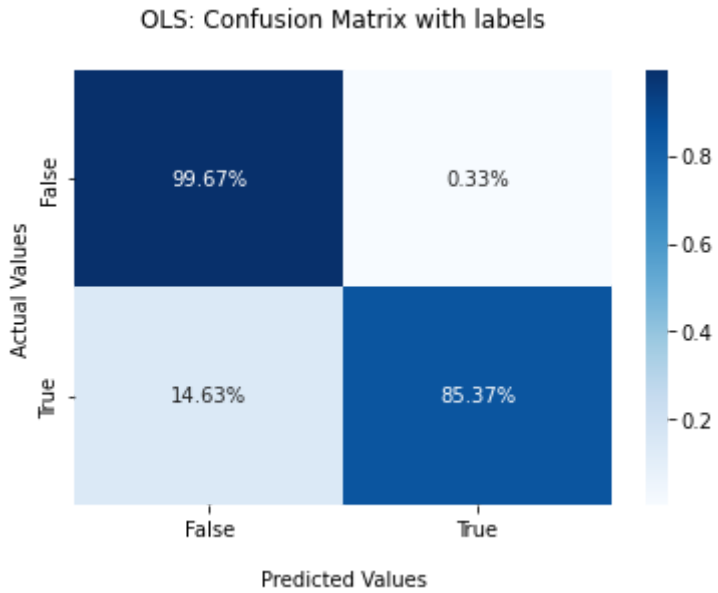
Out[9]:
```
0.9664082687338501
```

In [10]:
```
cm_knn1 = confusion_matrix(y_test, y_pred_knn1, normalize = 'true')
ax = sns.heatmap(cm_knn1, annot=True,
            fmt='.2%', cmap='Blues')

ax.set_title('OLS: Confusion Matrix with labels\n')
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values')

## Ticket Labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])

## Display the visualization of the Confusion Matrix.
plt.show()
```



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

In [11]:
```
fraction = sum((y_pred_knn1 == 1) & (y_test == 1))/sum(y_pred_knn1 == 1)

print(fraction)
```

```
0.9859154929577465
```

With firms that are predicted for tax evasion, a fraction of 98.6% actually evaded taxes.

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

In [12]:
```
fraction = sum((y_pred_knn1 == 1) & (y_test == 1))/sum(y_test == 1)
print(fraction)
```

```
0.8536585365853658
```

The fraction of the firms that evaded taxes which were predicted to have evaded taxes is 85.4%

5d) Determine whether KNN performs better with or without the attributes normalized

Normalized knn 1

In [13]:
```
#Normalized knn 1

x_norm = pd.DataFrame(preprocessing.scale(x))

x_norm_train = x_norm.head(388)
y_train = y.head(388)

x_norm_test = x_norm.tail(387)
y_test = y.tail(387)

knn_1 = KNeighborsClassifier(n_neighbors = 1)
knn_1.fit(x_norm_train, y_train)
y_pred_knn1 = knn_1.predict(x_norm_test)
accuracy_score(y_test, y_pred_knn1)
```

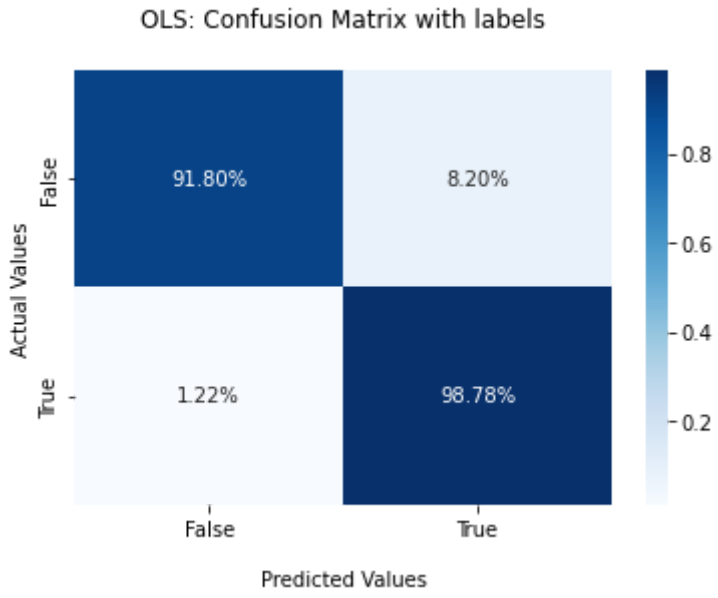
Out[13]:
```
0.9328165374677002
```

In [14]:
```
cm_knn2 = confusion_matrix(y_test, y_pred_knn1, normalize = 'true')
ax = sns.heatmap(cm_knn2, annot=True,
            fmt='.2%', cmap='Blues')

ax.set_title('OLS: Confusion Matrix with labels\n')
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values')

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])

## Display the visualization of the Confusion Matrix.
plt.show()
```



In [15]:
```
fraction = sum((y_pred_knn1 == 1) & (y_test == 1))/sum(y_pred_knn1 == 1)

print(fraction)
```

```
0.7641509433962265
```

With firms that are predicted for tax evasion, a fraction of 76.41% actually evaded taxes.

In [16]:
```
fraction = sum((y_pred_knn1 == 1) & (y_test == 1))/sum(y_test == 1)

print(fraction)
```

```
0.9878048780487805
```

The fraction of the firms that evaded taxes which were predicted to have evaded taxes is 98.7%

We have observed that, in our case, KNN provides a better accuracies when non-normalized. We observe that when non normalized the accuracy of the model comes out to be 96.64% and for normalized gives accuracy of 93.28%. Further explanation based on tradeoff (below).

However, we were able to predict more accurately at 98.7% with the firms that actually evaded taxes compared to just 85.4% when the attributes are non normalized. One of the possible reasons for this is when we have the attributes that are not normalized, the attributes whose units materially are larger than others can dominate the distance factors and result in a bias in classification. However when we normalize the attributes, the attributes whose units are materially larger than the others will no longer impact the distances calculated in KNN and thus significantly improve the performance of the model.

Question 6

(10 points) Repeat the previous question with k=5.

6(a) Construct the confusion matrix.

In [17]:
```
#No normalization

knn_5 = KNeighborsClassifier(n_neighbors = 5)
knn_5.fit(x_train, y_train)
```

```
y_pred_knn5 = knn_5.predict(x_test)
```

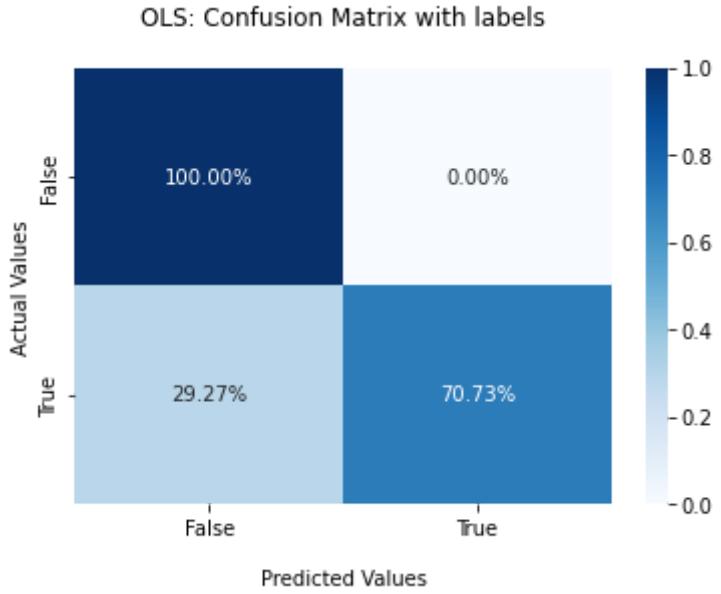
Out[17]:
```
0.937984496124031
```

In [18]:
```
cm_knn5 = confusion_matrix(y_test, y_pred_knn5, normalize = 'true')
ax = sns.heatmap(cm_knn5, annot=True,
            fmt='.2%', cmap='Blues')

ax.set_title('OLS: Confusion Matrix with labels\n')
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values')

## Ticket Labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])

## Display the visualization of the Confusion Matrix.
plt.show()
```



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

In [19]:
```
fraction = sum((y_pred_knn5 == 1) & (y_test == 1))/sum(y_pred_knn5 == 1)

print(fraction)
```

```
1.0
```

With firms that are predicted for tax evasion, a fraction of 100% actually evaded taxes.

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

In [20]:
```
fraction = sum((y_pred_knn5 == 1) & (y_test == 1))/sum(y_test == 1)

print(fraction)
```

```
0.7073170731707317
```

The fraction of the firms that evaded taxes were predicted to have evaded taxes is 70.73%

6d) Determine whether KNN performs better with or without the attributes normalized.

Normalized knn 5

In [21]:
```
#Normalized knn 5

x_norm = pd.DataFrame(preprocessing.scale(x))

x_norm_train = x_norm.head(388)
y_train = y.head(388)

x_norm_test = x_norm.tail(387)
y_test = y.tail(387)

knn_5 = KNeighborsClassifier(n_neighbors = 5)
knn_5.fit(x_norm_train, y_train)
y_pred_knn5 = knn_5.predict(x_norm_test)
accuracy_score(y_test, y_pred_knn5)
```

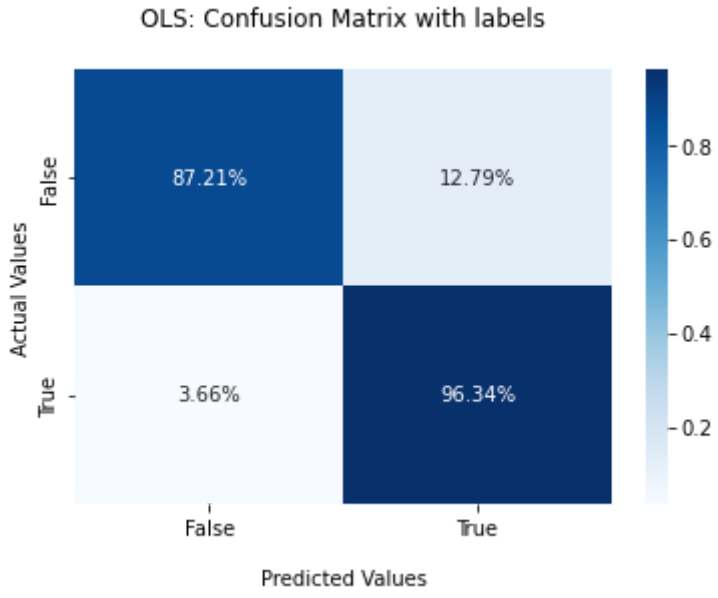
Out[21]:
```
0.8914728682170543
```

In [22]:
```
cm_knn5 = confusion_matrix(y_test, y_pred_knn5, normalize = 'true')
ax = sns.heatmap(cm_knn5, annot=True,
            fmt='.2%', cmap='Blues')

ax.set_title('OLS: Confusion Matrix with labels\n')
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values')

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])

## Display the visualization of the Confusion Matrix.
plt.show()
```



In [23]:
```
fraction = sum((y_pred_knn5 == 1) & (y_test == 1))/sum(y_pred_knn5 == 1)

print(fraction)
```

```
0.6694915254237288
```

With firms that are predicted for tax evasion, a fraction of 66.9% actually evaded taxes.

In [24]:
```
fraction = sum((y_pred_knn5 == 1) & (y_test == 1))/sum(y_test == 1)

print(fraction)
```

```
0.9634146341463414
```

The fraction of the firms that evaded taxes were predicted to have evaded taxes is 96.34%

We have observed that, in our case, KNN provides a better accuracies when non-normalized. We observe that when non normalized the accuracy of the model comes out to be 93.79% and for normalized gives accuracy of 89.14%. Further explanation based on tradeoff is below.

However, we were able to predict more accurately at 96.34% with the firms that actually evaded taxes compared to just 70.73% when the attributes are non normalized. One of the possible reasons for this is when we have the attributes that are not normalized, the attributes whose units materially are larger than others can dominate the distance factors and result in a bias in classification. However when we normalize the attributes, the attributes whose units are materially larger than the others will no longer impact the distances calculated in KNN and thus significantly improve the performance of the model.

Question 7

Which KNN model performs better? Briefly explain how you make this determination and why you think this is the case.

Here is what we have observed:

Non Normalized: KNN 1 model accuracy: 97% KNN 5 model accuracy: 94%

Normalized: KNN 1 model accuracy = 93% KNN 5 model accuracy = 89%

Based on the above scores, the KNN 1 model performs better in both when the attributes are non-normalized and normalized. In the non-normalized case, KNN 1 has an accuracy score of 97%, while KNN 5 has an accuracy score of 94%. In the normalized case, KNN 1 has an accuracy score of 93%, while KNN 5 has an accuracy score of 89%.

The difference in this accuracy scores suggests that a smaller k value of 1 may lead to overfitting while the k value of 5 may lead to underfitting. However this difference is only about 3% when the attributes are non normalized, and the difference is about 4% when the attributes are normalized. This difference though very small, needs to be given a thought on how it fits in the context. While accuracy can tell us a lot about the model performance, other metrics such as precision etc. needs to be considered as well.

We can also take the False positives and False negatives that arise in the models under consideration. And these too fall directly in relation to the context. A situation is called as false positive, when the model predicts that a firm has evaded taxes, but actually the firm has not evaded taxes. Such an error would cause the administration to investigate unnecessarily and increases the costs associated with audits.A situation is called false negative, when the model predicts that a firm has not evaded taxes, however actually the firm has evaded taxes. Such an error would allow the administration to miss out on the firms that are evading taxes and would result in a loss of taxes to the government.

Now to compare both the models, we need to understand the cost associated with both the situations. If the cumulative cost of auditing and going after a firm is higher than the tax losses, the False positive situations needs to be reduced, even if that means that we are letting go of more firms that are classified in false negative group. In such a case, when non normalized, KNN model with k= 5 would be better. When normalized attributes, KNN model where k = 1 would be better.

In the other case, if the tax losses are higher than the cumulative cost of auditing and going after the firms, then the Flase negative occurances needs to be reduced, even though that means that wrongly accusing and investigating the firms that are not evading taxes. In such a context, when non normalized, a KNN model with k = 1 would be a better model. When we use normalized attributes, a KNN model where k = 1 would be a better model.

We can also further consider finding the optimal k value for the given data set to determine the better model.

Question 8

8. (15 points) For KNN, which kyields the lowest error rate? By 5-fold cross-validation (5FCV), find the kwith the lowest classification error rate. (Use the entire dataset for 5FCV, shuffle the data randomly for splitting, and set random_state=13.)

In [25]:
```
def odd(n):
    return list(range(1, 2*n, 2))

ks = odd(194)
print(ks)
```

```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43,
45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 81, 83, 85,
87, 89, 91, 93, 95, 97, 99, 101, 103, 105, 107, 109, 111, 113, 115, 117, 119, 121, 1
23, 125, 127, 129, 131, 133, 135, 137, 139, 141, 143, 145, 147, 149, 151, 153, 155,
157, 159, 161, 163, 165, 167, 169, 171, 173, 175, 177, 179, 181, 183, 185, 187, 189,
191, 193, 195, 197, 199, 201, 203, 205, 207, 209, 211, 213, 215, 217, 219, 221, 223,
225, 227, 229, 231, 233, 235, 237, 239, 241, 243, 245, 247, 249, 251, 253, 255, 257,
259, 261, 263, 265, 267, 269, 271, 273, 275, 277, 279, 281, 283, 285, 287, 289, 291,
293, 295, 297, 299, 301, 303, 305, 307, 309, 311, 313, 315, 317, 319, 321, 323, 325,
327, 329, 331, 333, 335, 337, 339, 341, 343, 345, 347, 349, 351, 353, 355, 357, 359,
361, 363, 365, 367, 369, 371, 373, 375, 377, 379, 381, 383, 385, 387]
```

In [26]:
```
ac_rate = []
for i in ks:
     knn = KNeighborsClassifier(n_neighbors=i)
     knn.fit(x_norm_train, y_train)
     pred_i = knn.predict(x_norm_test)
     ac_rate.append(np.mean(pred_i == y_test))

max_value = max(ac_rate)

print(max_value)
```

```
0.9328165374677002
```

In [27]:
```
opt_k = ac_rate.index(max_value)
print(opt_k + 1)
```

```
1
```

For KNN, k = 1 yields the lowest error rate.

Trying cross validation technique

In [28]:
```
# We now try 5-fold cross-validation to estimate optimal k

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

```
{'n_neighbors': 1}
0.967741935483871
```

By 5-fold cross-validation (5FCV), again k = 1 gives the lowest error rate.

Question 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 qwhere you would "predict" the same number of firms as having committed tax fraud using the LPM.7 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.

In [29]:
```
# Optimal KNN Non-normalized

knn_1 = KNeighborsClassifier(n_neighbors = 1)
knn_1.fit(x_train, y_train)
y_pred_knn1 = knn_1.predict(x_test)
accuracy_score(y_test, y_pred_knn1)
```

Out[29]:
```
0.9664082687338501
```

In [30]:
```
#Number of firms that committed tax fraud
sum(y_pred_knn1)
```

Out[30]:
```
71
```

Accuracy score = 96.6%

Number of firms that committed tax fraud = 71

In [31]:
```
#Opt KNN 1

#Pool of firms that are classified as having evaded taxes
sum(y_pred_knn1)

#Firms that did not commit tax evasion.
sum((y_pred_knn1 == 1) & (y_test == 0))


proportion = sum((y_pred_knn1 == 1) & (y_test == 0))/sum(y_pred_knn1)
print(proportion)
```

```
0.014084507042253521
```

Then within the pool of firms that are classified as having evaded taxes, we identify the proportion of firms that did not commit tax evasion as 1.4% under Optimal KNN model.

In [32]:

```
#LPM
#Given number of firms that committed tax fraud = 71
decimals = np.arange(0, 1, 0.01)
decimals

result = []
lpm = LinearRegression()
lpm.fit(x_train, y_train)

for i in decimals:

    y_pred_prob = lpm.predict(x_test)
    y_pred = np.where(y_pred_prob > i, 1, 0)
    accuracy = (y_pred == y_test).mean()
    no_of_firms = sum(y_pred)
    result.append([i, accuracy.round(), no_of_firms])


print(result)
```

```
[[0.0, 1.0, 124], [0.01, 1.0, 124], [0.02, 1.0, 124], [0.03, 1.0, 124], [0.04, 1.0,
124], [0.05, 1.0, 124], [0.06, 1.0, 123], [0.07, 1.0, 123], [0.08, 1.0, 122], [0.09,
1.0, 117], [0.1, 1.0, 111], [0.11, 1.0, 107], [0.12, 1.0, 106], [0.13, 1.0, 104],
[0.14, 1.0, 103], [0.15, 1.0, 102], [0.16, 1.0, 101], [0.17, 1.0, 98], [0.18, 1.0, 9
6], [0.19, 1.0, 96], [0.2, 1.0, 94], [0.21, 1.0, 91], [0.22, 1.0, 90], [0.23, 1.0, 9
0], [0.24, 1.0, 89], [0.25, 1.0, 88], [0.26, 1.0, 82], [0.27, 1.0, 71], [0.28, 1.0,
64], [0.29, 1.0, 59], [0.3, 1.0, 53], [0.31, 1.0, 53], [0.32, 1.0, 51], [0.33, 1.0,
50], [0.34, 1.0, 49], [0.35000000000000003, 1.0, 48], [0.36, 1.0, 46], [0.37, 1.0, 4
5], [0.38, 1.0, 44], [0.39, 1.0, 44], [0.4, 1.0, 44], [0.41000000000000003, 1.0, 4
4], [0.42, 1.0, 44], [0.43, 1.0, 38], [0.44, 1.0, 37], [0.45, 1.0, 35], [0.46, 1.0,
34], [0.47000000000000003, 1.0, 33], [0.48, 1.0, 33], [0.49, 1.0, 33], [0.5, 1.0, 3
3], [0.51, 1.0, 31], [0.52, 1.0, 30], [0.53, 1.0, 28], [0.54, 1.0, 28], [0.55, 1.0,
27], [0.56, 1.0, 25], [0.5700000000000001, 1.0, 22], [0.58, 1.0, 21], [0.59, 1.0, 2
0], [0.6, 1.0, 19], [0.61, 1.0, 19], [0.62, 1.0, 19], [0.63, 1.0, 18], [0.64, 1.0, 1
8], [0.65, 1.0, 16], [0.66, 1.0, 16], [0.67, 1.0, 14], [0.68, 1.0, 14], [0.690000000
0000001, 1.0, 14], [0.7000000000000001, 1.0, 14], [0.71, 1.0, 14], [0.72, 1.0, 13],
[0.73, 1.0, 13], [0.74, 1.0, 13], [0.75, 1.0, 13], [0.76, 1.0, 13], [0.77, 1.0, 13],
[0.78, 1.0, 12], [0.79, 1.0, 11], [0.8, 1.0, 11], [0.81, 1.0, 11], [0.820000000000000
01, 1.0, 11], [0.8300000000000001, 1.0, 11], [0.84, 1.0, 11], [0.85, 1.0, 11], [0.8
6, 1.0, 11], [0.87, 1.0, 10], [0.88, 1.0, 10], [0.89, 1.0, 10], [0.9, 1.0, 9], [0.9
1, 1.0, 8], [0.92, 1.0, 7], [0.93, 1.0, 6], [0.9400000000000001, 1.0, 6], [0.9500000
000000001, 1.0, 6], [0.96, 1.0, 6], [0.97, 1.0, 6], [0.98, 1.0, 6], [0.99, 1.0, 6]]
```

We see that at a treshold 0.27 we find that LPM predicts the same number of firms to fraud as predicted by the optimal KNN. The accuracy at this treshold is approximately 100%.

In [33]:

```
#LPM

y_pred_prob = lpm.predict(x_test)
y_pred = np.where(y_pred_prob > 0.27, 1, 0)



#Pool of firms that are classified as having evaded taxes
sum(y_pred_prob)

#Firms that did not commit tax evasion.
```

```
sum((y_pred_prob == 1) & (y_test == 0))


proportion = sum((y_pred == 1) & (y_test == 0))/sum(y_pred)
print(proportion)
```

```
0.3380281690140845
```

Then within the pool of firms that are classified as having evaded taxes, we identify the proportion of firms that did not commit tax evasion as 33.8% under LPM model.

This shows that the false positives where the firms who actully did not commit tax evasion are classified by the model as evading taxes is higher in LPM and lower in Optimal KNN model. This deems that LPM wrongly accuses about 33% of the firms who actually did not commit any tax evasion as evading taxes. However, the optimal KNN wrongly accuses only 1.4%(very less compared to the LPM model) of the firms who actually did not commit any tax evasion as evading taxes.

Thus KNN model(k = 1) would be a better model if the context is that false positive is an important measure for the government, which means that having auditing more firms is a cost and reputational concern for the government. The KNN model results in less audits as it identifies less firms as tax evaders even when they are not compared to what the LPM model identifies.

Question 10

10. (5 points) 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? Hint: the firms in the data are all firms that were audited.

In the long run, selection bias is the problem that might arise from the nature of the sample if the government heavily uses our best KNN model to target audits. The reason being that the firms in the sample when the government heavily uses our best KNN model will be all the firms that were audieted. Such a sample may not accurately represent the overall population of the firms in the market. Thus selection bias. This can result in wrong assumptions of modeling, eventually resulting in lesser accurate model for the overall population. Using the data that is affected with selection bias can also result in over-fitting, where the model would not work well on unseen data. This is again due to the wrong understanding or assumptions arisen from the sample we used.

This could also result in excessive auditing by the government, even of the firms that are actually very less likely to evade taxes. Moreover, it could also result in legal, political, privacy violations and can put the government in jeopardy.

In [ ]: