MP02

Machine Learning - Python Programming

Answers Key

```
from google.colab import drive
drive.mount('/content/drive/')
```

```
     Mounted at /content/drive/
```

```
# Read Data
import pandas as pd
import numpy as np
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV, KFold
from sklearn.linear_model import LinearRegression

data = pd.read_csv("/content/drive/My Drive/23-winter-ml-python/Data-Audit.csv")
```

2 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.*

This question is very open. In order to obtain full credit for this question, you must provide good reasoning behind which variables present or not present in the dataset might affect the true model. You must also provide arguments for more than one variable.

Some examples from the variables in the dataset:

- Sector: A firm might be more likely to evade taxes if this is a common practice within the sector (positive correlation).
- Score: The higher the calculated score, higher the probability of evading taxes (positive correlation).
- Inherent risk: The higher historical score, higher the probability of evading taxes (positive correlation).

The range of variables that you can mention that are not in the dataset is extensive. Some examples:

- Tax rates
- Corruption in the country where the firm is based
- Political variables

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?*

The main advantage of KNN over LPM is the fact that KNN is a non-parametric model, which doesn't require knowing the real functional form of X as the model does not make any assumptions regarding the distribution of the data.

The omission of an interaction term (which belongs to the true model) when specifying our LPM induces omitted variable bias, leading to an erroneous interpretation of our estimated coefficients.

In KNN, we don't need the specification of the interaction, as KNN works with the distance between our target and a K number of points, predicting according to the most repeated pattern in that space.

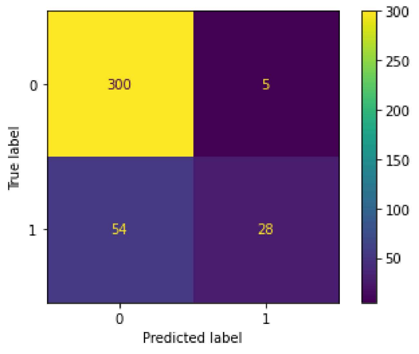
```
# Transform and split data
data = data.dropna()

train = data.head(388)
test = data.tail(387)
x_train = train.drop(['Risk'], axis = 1)
y_train = train['Risk']
x_test = test.drop(['Risk'], axis = 1)
y_test = test['Risk']
```

```
# Build model
model = LinearRegression()
lpm = model.fit(x_train, y_train)
y_pred_lpm = model.predict(x_test)
```

Question 3a)

```
y_pred_lpm_cat = np.where(y_pred_lpm > 0.5, 1, 0)
cm_lpm = confusion_matrix(y_test, y_pred_lpm_cat)
ConfusionMatrixDisplay(cm_lpm).plot()
```

```
    <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f9887766e80>
```

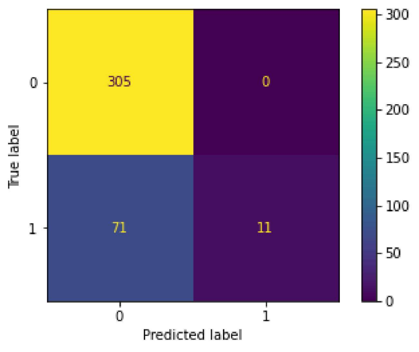


Solution: 28/33

Question 3b)

```
y_pred_lpm_cat = np.where(y_pred_lpm > 0.8, 1, 0)
cm_lpm = confusion_matrix(y_test, y_pred_lpm_cat)
ConfusionMatrixDisplay(cm_lpm).plot()
```

```
    <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f988252a820>
```



Solution: 11/11

Question 3c) - See confusion matrixes above

Question 4a:

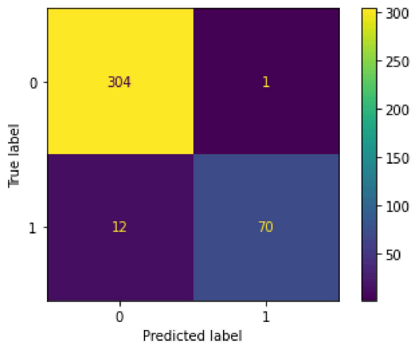
Full points would have been given with any reasonable discussion about the real-world costs and tradeoffs between optimizing for false negative rates and optimizing for false positive rates. For example, if we think that auditing a company that is not evading taxes is less problematic than not auditing a company that evades taxes we would not worry about false positives as much as false negatives. This question required discussing the possible costs we incur or the incentives we create by optimizing for false positives or negatives.

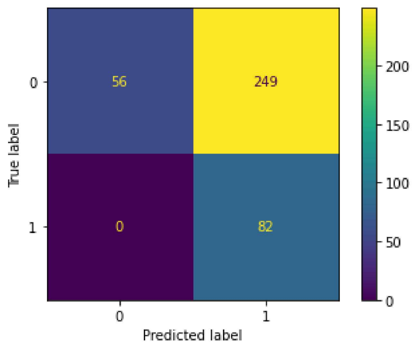
Question 5

Solution 5a: Confusion matrices (without and with normalization)

```
x = data.drop(['Risk'], axis = 1)
y = data['Risk']
x = x.dropna()

# Without normalizing data attributes
x_train = x.head(388)
x_test = x.tail(387)

y_train = y.head(388)
```

```
y_test = y.tail(387)

knn_1_nonorm = KNeighborsClassifier(n_neighbors = 1)
knn_1_nonorm.fit(x_train, y_train)
y_pred_knn1_nonorm = knn_1_nonorm.predict(x_test)
cm_knn1_nonorm = confusion_matrix(y_test, y_pred_knn1_nonorm)

ConfusionMatrixDisplay(cm_knn1_nonorm).plot()

# Normalize our splitted data
x_train_norm = pd.DataFrame(preprocessing.scale(x_train))
x_test_norm = pd.DataFrame(preprocessing.scale(x_test))

# Fit Model
knn_1_norm = KNeighborsClassifier(n_neighbors = 1)
knn_1_norm.fit(x_train_norm, y_train)
y_pred_knn1_norm = knn_1_norm.predict(x_test_norm)
cm_knn1_norm = confusion_matrix(y_test, y_pred_knn1_norm)

ConfusionMatrixDisplay(cm_knn1_norm).plot()
     <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f9881f13fd0>
```





Solution 5b:

Without the predictors normalized, 70/71 (98.59%) of firms that are predicted for tax evasion have actually evaded taxes

With the predictors normalized, 82/331 (24.77%) of firms that are predicted for tax evasion have actually evaded taxes

Solution 5c:

Without the predictors normalized, 70/82 (85.37%) of firms that committed tax evasion are predicted to have done so.

With the predictors normalized, 82/82 (100%) of firms that committed tax evasion are predicted to have done so.

Solution 5d:

Without normalization, k-NN catches far fewer false positives (1 vs 249). But with normalization, k-NN catches far more fraudulent firms (82 vs 70). Unless the cost of an audit is drastically low relative to the financial gain of auditing a firm that evaded taxes (from recuperating taxes owed as well as receiving fines), k-NN without normalization appears preferrable.

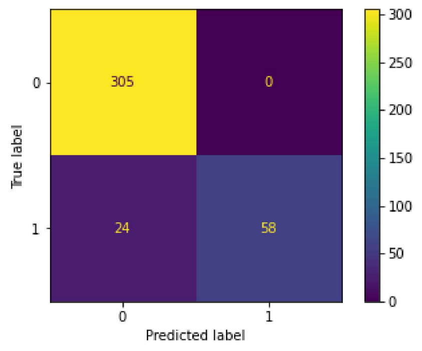
Question 6

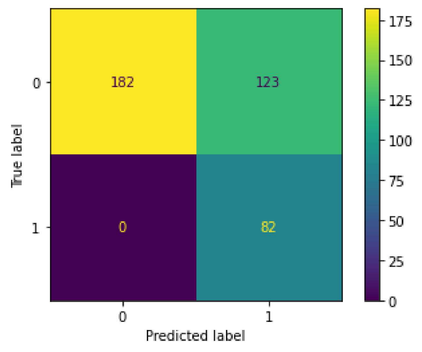
Solution 6a: Confusion matrices (without and with normalization)

```
knn_5_nonorm = KNeighborsClassifier(n_neighbors = 5)
knn_5_nonorm.fit(x_train, y_train)
y_pred_knn5_nonorm = knn_5_nonorm.predict(x_test)
cm_knn5_nonorm = confusion_matrix(y_test, y_pred_knn5_nonorm)

ConfusionMatrixDisplay(cm_knn5_nonorm).plot()
```

```
# Fit Model
knn_5_norm = KNeighborsClassifier(n_neighbors = 5)
knn_5_norm.fit(x_train_norm, y_train)
y_pred_knn5_norm = knn_5_norm.predict(x_test_norm)
cm_knn5_norm = confusion_matrix(y_test, y_pred_knn5_norm)

ConfusionMatrixDisplay(cm_knn5_norm).plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f9881d3fd90>
```





Solution 6b:

Without the predictors normalized, 58/58 (100%) of firms that are predicted for tax evasion have actually evaded taxes

With the predictors normalized, 82/205 (40.00%) of firms that are predicted for tax evasion have actually evaded taxes

Solution 6c:

Without the predictors normalized, 58/82 (70.73%) of firms that committed tax evasion are predicted to have done so.

With the predictors normalized, 82/82 (100%) of firms that committed tax evasion are predicted to have done so.

Solution 6d:

Without normalization, k-NN catches far fewer false positives (0 vs 123). But with normalization, k-NN catches several more fraudulent firms (82 vs 58). Unless the expected cost of 123 audits is lower than the expected financial gain of auditing 24 more firms that evaded taxes (from recuperating taxed owed as well as receiving fines), k-NN without normalization appears preferrable.

Question 7

In a perfect world, we'd want a model that has a accuracy of 1, a precision (TP/(TP+FP)) of 1 and a recall (TP/(TP+FN)) of 1. That means a F1-score ((2*TP)/(2*TP + FP + FN)) of 1, i.e. a 100% accuracy which is often not the case. So what we should try, is to get a higher precision with a higher recall value.

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

1. K = 1, attributes not normalized

```
print('\nAccuracy: {:.2f}\n'.format(accuracy_score(y_test, y_pred_knn1_nonorm)))
print('Precision: {:.2f}'.format(precision_score(y_test, y_pred_knn1_nonorm)))
print('Recall: {:.2f}'.format(recall_score(y_test, y_pred_knn1_nonorm)))
print('F1-score: {:.2f}\n'.format(f1_score(y_test, y_pred_knn1_nonorm)))
```

```
Accuracy: 0.97

Precision: 0.99
```

```
     Recall: 0.85
     F1-score: 0.92
```

2. K = 1, attributes normalized

```
print('\nAccuracy: {:.2f}\n'.format(accuracy_score(y_test, y_pred_knn1_norm)))
print('Precision: {:.2f}'.format(precision_score(y_test, y_pred_knn1_norm)))
print('Recall: {:.2f}'.format(recall_score(y_test, y_pred_knn1_norm)))
print('F1-score: {:.2f}\n'.format(f1_score(y_test, y_pred_knn1_norm)))
```

```
     Accuracy: 0.36

     Precision: 0.25
     Recall: 1.00
     F1-score: 0.40
```

3. K = 5, attributes not normalized

```
print('\nAccuracy: {:.2f}\n'.format(accuracy_score(y_test, y_pred_knn5_nonorm)))
print('Precision: {:.2f}'.format(precision_score(y_test, y_pred_knn5_nonorm)))
print('Recall: {:.2f}'.format(recall_score(y_test, y_pred_knn5_nonorm)))
print('F1-score: {:.2f}\n'.format(f1_score(y_test, y_pred_knn5_nonorm)))
```

```
     Accuracy: 0.94

     Precision: 1.00
     Recall: 0.71
     F1-score: 0.83
```

4. K = 5, attributes normalized

```
print('\nAccuracy: {:.2f}\n'.format(accuracy_score(y_test, y_pred_knn5_norm)))
print('Precision: {:.2f}'.format(precision_score(y_test, y_pred_knn5_norm)))
print('Recall: {:.2f}'.format(recall_score(y_test, y_pred_knn5_norm)))
print('F1-score: {:.2f}\n'.format(f1_score(y_test, y_pred_knn5_norm)))
```

```
     Accuracy: 0.68

     Precision: 0.40
     Recall: 1.00
     F1-score: 0.57
```

Based on the above analysis, k-NN model with k = 1 and the attributes not normalized, performs better as compared to all other models

Question 8.

```
# Create new model
knn = KNeighborsClassifier()

# Create dictionary with grid values to pass to GridSearchCV
grid = {'n_neighbors': list(range(1, 2*194, 2))}

knn_5fcv = GridSearchCV(estimator=knn,
                      param_grid=grid,
                      cv = KFold(5, random_state=13, shuffle=True))

# Fit model
knn_5fcv.fit(x, y)
```

```
     GridSearchCV(cv=KFold(n_splits=5, random_state=13, shuffle=True),
                  estimator=KNeighborsClassifier(),
                  param_grid={'n_neighbors': [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, ...]})
```

```
print(f"k that yields the lowest error rate is k={knn_5fcv.best_params_['n_neighbors']},"
f" and lowest error rate is {round(1-knn_5fcv.best_score_, 4)}.")
```

```
k that yields the lowest error rate is k=1, and lowest error rate is 0.0206.
```

Question 9.

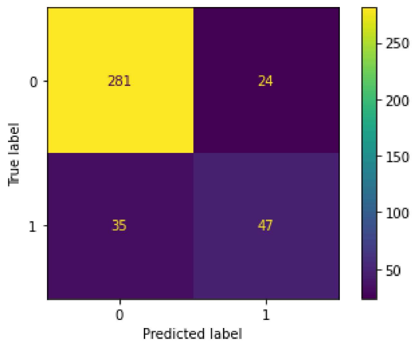
From 5(a) we know that for the optimal KNN, the number of firms that are predicted to have commited tax evasion is 71. Then, we will find the threshold q that will get us the same number of predicted tax evaders using LPM.

```
model = LinearRegression()
lpm = model.fit(x_train, y_train)
y_pred_lpm = model.predict(x_test)
target = 71


for threshold in range(0, 50, 1):
  y_pred_lpm_cat = np.where(y_pred_lpm > (threshold/100), 1, 0)
  if np.count_nonzero(y_pred_lpm_cat) == target:
    q = threshold/100
    break

print(f"Threshold that results in a prediction of {target} firms"
f" having commited tax evasion is q = {q}.")
```

```
Threshold that results in a prediction of 71 firms having commited tax evasion is q = 0.27.
```

```
cm_lpm = confusion_matrix(y_test, y_pred_lpm_cat)
ConfusionMatrixDisplay(cm_lpm).plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f9881d3f7c0>
```



From the confusion matrix displayed above we see that the False Positive Rate for LPM with a threshold q= 0.27 is 33.80% (24/71). In 5(a) we found that the False Positive Rate for KNN model with k = 1 is 1.41% (1/71). Thus, KNN model with k = 1 results in a lower FPR.

Question 10.

Open-ended question. Full points if response covers the following points.

- Dataset only includes firms that the goverment initially suspected that were evading taxes. Firms that were good at tax evasion are not in the data.
- KNN model was trained only with the firms that are not great at tax evasion, as a result the model might fail to catch firms that are good tax evaders.
- In the long run, other firms might imitate the practices of the firms that are good at tax evasion.