hw4

April 24, 2024

[22]:
```
# Forest-for-the-Trees Questions
# 2.1
# Several factors influence whether a firm commits tax evasion. Financial␣
↪indicators like revenue, expenditure, and profit margin reflect its␣
↪financial health and likelihood of evasion. Industry and sector regulations␣
↪also play a role, with complex structures in larger firms providing more␣
↪evasion opportunities. Compliance history, economic conditions, and legal␣
↪changes further shape evasion decisions.
```

[23]:
```
# 2.2
# If the true model includes an interaction between predictors not explicitly␣
↪in the fitted model, KNN may outperform LPM. KNN's proximity-based approach␣
↪captures interactions implicitly, while LPM, assuming linearity, may miss␣
↪complex interactions. Thus, KNN's ability to discern nonlinear relationships␣
↪makes it better suited for detecting crucial interactions not explicitly␣
↪modeled.
```

[24]:
```
# 3
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import confusion_matrix, roc_curve, auc
import matplotlib.pyplot as plt

data = pd.read_csv("/Users/rouren/Desktop/24S ML/HW/hw4/Data-Audit.csv")
data.dropna(inplace=True)

# Split the dataset into training and validation sets
train, validation = train_test_split(data, test_size=0.5, random_state=13)

# Separate predictors and target variable
X_train = train.drop(columns=['Risk'])
y_train = train['Risk']
X_validation = validation.drop(columns=['Risk'])
y_validation = validation['Risk']

# Fit a LPM
lpm_model = LinearRegression()
```

```
lpm_model.fit(X_train, y_train)

predicted_probabilities = lpm_model.predict(X_validation)
```

[25]:
```
# (a)
threshold_05_predictions = (predicted_probabilities > 0.5).astype(int)
conf_matrix_05 = confusion_matrix(y_validation, threshold_05_predictions)
print("Confusion Matrix (Threshold > 0.5):\n", conf_matrix_05)
```

```
Confusion Matrix (Threshold > 0.5):
 [[221   8]
 [ 29 130]]
```

[26]:
```
# (b)
threshold_06_predictions = (predicted_probabilities > 0.6).astype(int)
conf_matrix_06 = confusion_matrix(y_validation, threshold_06_predictions)
print("Confusion Matrix (Threshold > 0.6):\n", conf_matrix_06)
```

```
Confusion Matrix (Threshold > 0.6):
 [[225   4]
 [ 39 120]]
```

[27]:
```
# (c)
error_rate_05 = (conf_matrix_05[0, 1] + conf_matrix_05[1, 0]) /␣
 ↪len(y_validation)
error_rate_06 = (conf_matrix_06[0, 1] + conf_matrix_06[1, 0]) /␣
 ↪len(y_validation)
print("Error Rate (Threshold > 0.5):", error_rate_05)
print("Error Rate (Threshold > 0.6):", error_rate_06)
```

```
Error Rate (Threshold > 0.5): 0.09536082474226804
Error Rate (Threshold > 0.6): 0.11082474226804123
```

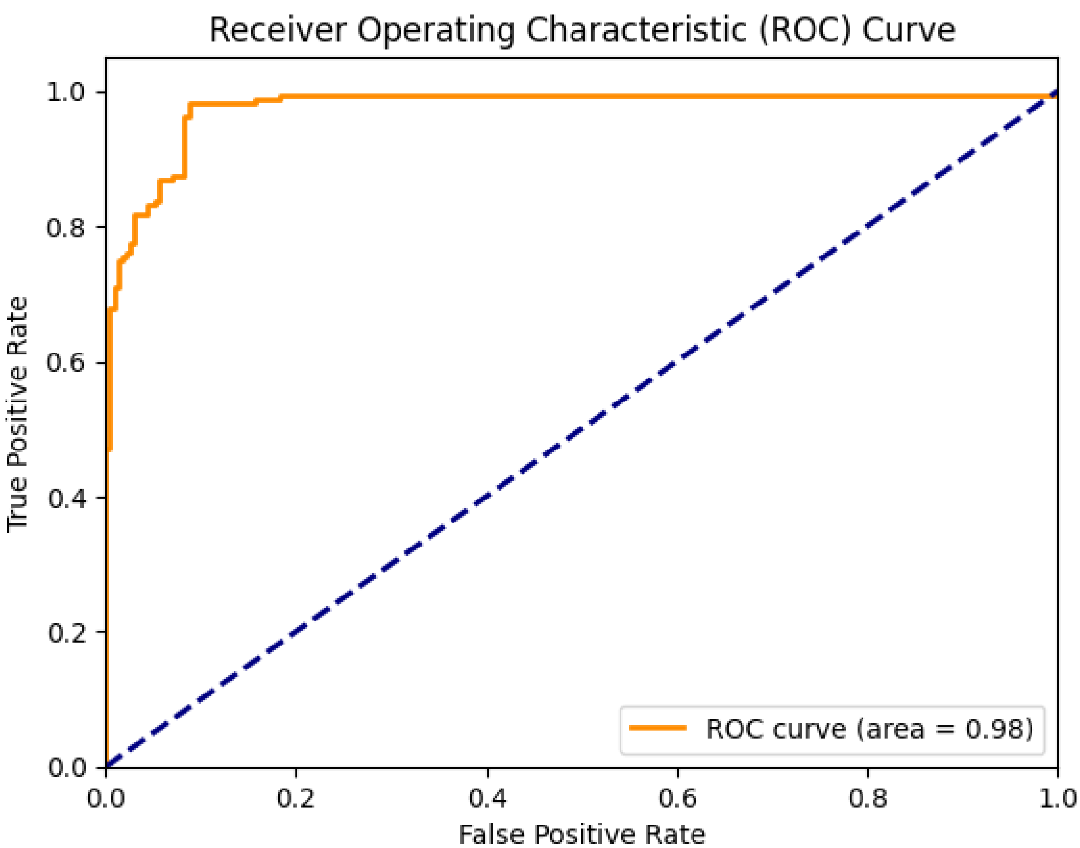
[28]:
```
# (d)
prop_actual_evasion_06 = conf_matrix_06[1, 1] / (conf_matrix_06[1, 1] +␣
 ↪conf_matrix_06[0, 1])
print("Proportion of Actual Tax Evasion (Threshold > 0.5):",␣
 ↪prop_actual_evasion_05)
print("Proportion of Actual Tax Evasion (Threshold > 0.6):",␣
 ↪prop_actual_evasion_06)
```

```
Proportion of Actual Tax Evasion (Threshold > 0.5): 0.9420289855072463
Proportion of Actual Tax Evasion (Threshold > 0.6): 0.967741935483871
```

[29]:
```
# (e)
fpr, tpr, thresholds = roc_curve(y_validation, predicted_probabilities)
roc_auc = auc(fpr, tpr)

plt.figure()
```

```
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' %␣
↪roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```



[30]:
```
# 4
# In performance evaluation for predicting tax evasion, false negatives are␣
↪more significant than false positives. Adjusting the classification␣
↪threshold affects this balance: lowering it increases sensitivity but may␣
↪raise false positives, while raising it reduces false positives but may␣
↪increase false negatives. The choice depends on the trade-off between missed␣
↪revenue and unnecessary interventions on compliant firms.
```

[31]:
```
# 5
from sklearn.neighbors import KNeighborsClassifier

knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train, y_train)

knn_predicted_probabilities = knn_model.predict_proba(X_validation)[:, 1]
knn_threshold_predictions = (knn_predicted_probabilities > 0.5).astype(int)
```

[32]:
```
# (a)
conf_matrix_knn = confusion_matrix(y_validation, knn_threshold_predictions)
print("Confusion Matrix (KNN):\n", conf_matrix_knn)
```

```
Confusion Matrix (KNN):
 [[226   3]
 [ 11 148]]
```

[33]:
```
# (b)
error_rate_knn = (conf_matrix_knn[0, 1] + conf_matrix_knn[1, 0]) /␣
 ↪len(y_validation)
print("Error Rate (KNN):", error_rate_knn)
```

```
Error Rate (KNN): 0.03608247422680412
```

[34]:
```
# (c)
prop_actual_evasion_knn = conf_matrix_knn[1, 1] / (conf_matrix_knn[1, 1] +␣
 ↪conf_matrix_knn[0, 1])
print("Proportion of Actual Tax Evasion (KNN):", prop_actual_evasion_knn)
```

```
Proportion of Actual Tax Evasion (KNN): 0.9801324503311258
```

[35]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_validation_scaled = scaler.transform(X_validation)

knn_model_scaled = KNeighborsClassifier(n_neighbors=5)
knn_model_scaled.fit(X_train_scaled, y_train)

knn_predicted_probabilities_scaled = knn_model_scaled.
 ↪predict_proba(X_validation_scaled)[:, 1]
knn_threshold_predictions_scaled = (knn_predicted_probabilities_scaled > 0.5).
 ↪astype(int)
```

[36]:
```
# (a)
conf_matrix_knn_scaled = confusion_matrix(y_validation,␣
 ↪knn_threshold_predictions_scaled)
```

```
print("Confusion Matrix (KNN with scaled predictors):\n",␣
↪conf_matrix_knn_scaled)
```

```
Confusion Matrix (KNN with scaled predictors):
 [[221   8]
 [ 12 147]]
```

[37]:
```
# (b)
error_rate_knn_scaled = (conf_matrix_knn_scaled[0, 1] +␣
↪conf_matrix_knn_scaled[1, 0]) / len(y_validation)
print("Error Rate (KNN with scaled predictors):", error_rate_knn_scaled)
```

```
Error Rate (KNN with scaled predictors): 0.05154639175257732
```

[38]:
```
# (c)
prop_actual_evasion_knn_scaled = conf_matrix_knn_scaled[1, 1] /␣
↪(conf_matrix_knn_scaled[1, 1] + conf_matrix_knn_scaled[0, 1])
print("Proportion of Actual Tax Evasion (KNN with scaled predictors):",␣
↪prop_actual_evasion_knn_scaled)
```

```
Proportion of Actual Tax Evasion (KNN with scaled predictors):
0.9483870967741935
```

[39]:
```
# 7
# The KNN model without normalization performs better. It has a lower error␣
↪rate (0.036 vs. 0.052) and a higher proportion of actual tax evasion␣
↪correctly identified (0.980 vs. 0.948). Normalization may not have␣
↪significantly improved performance as the features might not have had widely␣
↪different scales or the distance metric used in KNN may not have been␣
↪significantly affected by feature scale.
```

[40]:
```
# 8
from sklearn.model_selection import cross_val_score

k_values = range(1, 21)

error_rates = {}

for k in k_values:
    knn_model_cv = KNeighborsClassifier(n_neighbors=k)
    cv_scores = cross_val_score(knn_model_cv, X_train, y_train, cv=5,␣
↪scoring='accuracy')
    error_rates[k] = 1 - cv_scores.mean()  # Classification error rate

best_k = min(error_rates, key=error_rates.get)
lowest_error_rate = error_rates[best_k]

print("The k value with the lowest error rate:", best_k)
```

```
print("Lowest error rate:", lowest_error_rate)
```

```
The k value with the lowest error rate: 1
Lowest error rate: 0.036030636030636054
```

[41]:
```
# The value of k that yields the lowest error rate is 1, with a lowest error rate of approximately 0.036.
# This result suggests that using only the nearest neighbor for classification leads to the lowest error rate in this particular dataset and model setup. However, it's essential to consider potential overfitting with such a small value of k, as using only one neighbor can make the model highly sensitive to noise in the data.
```

[42]:
```
# Relying heavily on the best KNN model to target audits based on the available sample might introduce sampling bias. Since the dataset includes only audited firms, the model's predictions may disproportionately target firms with similar characteristics, potentially overlooking other firms at risk of tax evasion. This could lead to inefficiencies in tax collection and inequalities in the tax system.
```