cmps320_hw4_SammanBhetwal

November 11, 2024

0.1 Part A

0.1.1 Question 1:

The model is overfitted if it performs well on the training data but generalizes poorly to new instances.It means the model has memorized the training data instead of understanding the pattern.

Two possible solutions to solve this problem are:

(i) Cross-validation: Cross-validation is a technique used to estimate the test error of a statistical learning method, helping assess its performance and determine the optimal level of model flexibility. In this approach, the dataset is divided into multiple folds. The model is repeatedly trained on some of these folds and validated on others, ensuring consistent performance across different subsets of the dataset.

(ii) Regularization: This approach involves fitting a model with all predictors, but the estimated coefficients are shrunk towards zero compared to the least squares estimates. This technique reduces variance by adding a penalty for model complexity, which helps minimize overfitting.

0.1.2 Question 2:

$$N_1 = 5 \tag{1}$$

$$N_2 = \left(\frac{200}{5}\right) \times (5-1) = 40 \times 4 = 160 \tag{2}$$

$$N_3 = \frac{200}{5} = 40 \tag{3}$$

0.1.3 Question 3

$$\text{TP} = 970 \tag{4}$$

$$\text{FN} = 25 \tag{5}$$

$$\text{TN} = \text{I5} \tag{6}$$

$$\text{FP} = 10 \tag{7}$$

$$\text{Total} = \text{TP} + \text{FN} + \text{TN} + \text{FP} = 1020 \tag{8}$$

(a)

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total}} = \frac{970 + 15}{1020} = 0.9657 \tag{9}$$

$$\text{Accuracy} = 96.57\% \tag{10}$$

(b)

$$\text{Majority-Class Baseline Accuracy} = \frac{\text{TP} + \text{FN}}{\text{Total}} = \frac{970 + 25}{1020} = \frac{995}{1020} = 0.9755 \tag{11}$$

$$\text{Majority-Class Baseline Accuracy} = 97.55\% \tag{12}$$

(c) The classifier has a lower accuracy than the majority class baseline, which might suggest it is less effective than simply predicting the majority class. However, since the dataset is imbalanced, accuracy alone is not a sufficient metric to judge the model's performance. We should also consider other metrics, such as precision and recall. By comparing these additional metrics, we can better assess the model's usefulness relative to the baseline.

(d)

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{970}{970 + 10} = 0.9898 \tag{13}$$

$$\text{Precision} = 98.98\% \tag{14}$$

(e)

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{970}{970 + 25} = \frac{970}{995} = 0.9749 \tag{15}$$

$$\text{Recall} = 97.49\% \tag{16}$$

(f)

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = 2 \times \frac{0.9898 \times 0.9749}{0.9898 + 0.9749} = 0.9823 \tag{17}$$

$$\text{F1-Score} = 98.23\% \tag{18}$$

0.2 Part B

[62]:
```
#importing necessary modules
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import precision_score, recall_score,␣
 ↪f1_score,confusion_matrix
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

1. Understanding the Dataset: Given the provided datasets (as CSV files), load them and answer the following questions.

(a) What is the dimension of the datasets

[50]:
```
caravan = pd.read_csv("Caravan.csv")
caravan.head(10)
```

[50]:

	Unnamed: 0	MOSTYPE	MAANTHUI	MGEMOMV	MGEMLEEF	MOSHOOFD	MGODRK	MGODPR	\
0	1	33	1	3	2	8	0	5	
1	2	37	1	2	2	8	1	4	
2	3	37	1	2	2	8	0	4	
3	4	9	1	3	3	3	2	3	
4	5	40	1	4	2	10	1	4	
5	6	23	1	2	1	5	0	5	
6	7	39	2	3	2	9	2	2	
7	8	33	1	2	3	8	0	7	
8	9	33	1	2	4	8	0	1	
9	10	11	2	3	3	3	3	5	

	MGODOV	MGODGE	…	APERSONG	AGEZONG	AWAOREG	ABRAND	AZEILPL	APLEZIER	\
0	1	3	…	0	0	0	1	0	0	
1	1	4	…	0	0	0	1	0	0	
2	2	4	…	0	0	0	1	0	0	
3	2	4	…	0	0	0	1	0	0	
4	1	4	…	0	0	0	1	0	0	
5	0	5	…	0	0	0	0	0	0	
6	0	5	…	0	0	0	0	0	0	
7	0	2	…	0	0	0	0	0	0	
8	3	6	…	0	0	0	0	0	0	
9	0	2	…	0	0	0	1	0	0	

	AFIETS	AINBOED	ABYSTAND	Purchase
0	0	0	0	No
1	0	0	0	No
2	0	0	0	No
3	0	0	0	No
4	0	0	0	No
5	0	0	0	No
6	0	0	0	No
7	0	0	0	No
8	0	0	0	No
9	0	0	0	No

[10 rows x 87 columns]

[51]:
```
caravan.shape
```

[51]: (5822, 87)

The dimension of the dataset is (5822, 87)

(b) How many predictors measure demographic characteristics?

[52]:
```
predictors_count = sum(caravan.columns.str.startswith('M'))
print(predictors_count)
```

```
43
```

It seems 43 predictors meausure demographic chatacteristics

(c) What is the percentage of people who purchased caravan insurance?

[53]:
```
total_caravan_records = len(caravan)
total_caravan_purchase = (caravan['Purchase'] == 'Yes').sum()
print(round((total_caravan_purchase / total_caravan_records) * 100, 2))
```

```
5.98
```

5.98% of the people purchased caravan insurance

2. Data preprocessing: Standardize the data matrix X, giving all variables a mean of zero and a standard deviation of one. In standardizing the datasets, exclude the response variable.

[54]:
```
X = caravan.drop(['Purchase', 'Unnamed: 0'], axis=1)
# Dropping 'Unnamed: 0' as it seems to be an index
```

[55]:
```
mean_values = X.mean()
standard_devs = X.std()
(mean_values, standard_devs)
```

[55]:
```
(MOSTYPE     24.253349
 MAANTHUI     1.110615
 MGEMOMV      2.678805
 MGEMLEEF     2.991240
 MOSHOOFD     5.773617
               ...
 AZEILPL      0.000515
 APLEZIER     0.006012
 AFIETS       0.031776
 AINBOED      0.007901
 ABYSTAND     0.014256
 Length: 85, dtype: float64,
 MOSTYPE     12.846706
 MAANTHUI     0.405842
 MGEMOMV      0.789835
 MGEMLEEF     0.814589
 MOSHOOFD     2.856760
               ...
 AZEILPL      0.022696
 APLEZIER     0.081632
 AFIETS       0.210986
 AINBOED      0.090463
 ABYSTAND     0.119996
```

```
Length: 85, dtype: float64)
```

[56]:
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled_df = pd.DataFrame(X_scaled, columns=X.columns)
mean_values = X_scaled_df.mean().round(2)
standard_devs = X_scaled_df.std().round(2)
(mean_values, standard_devs)
```

[56]:
```
(MOSTYPE    -0.0
 MAANTHUI   -0.0
 MGEMOMV     0.0
 MGEMLEEF    0.0
 MOSHOOFD    0.0
            …
 AZEILPL    -0.0
 APLEZIER   -0.0
 AFIETS     -0.0
 AINBOED     0.0
 ABYSTAND   -0.0
 Length: 85, dtype: float64,
 MOSTYPE     1.0
 MAANTHUI    1.0
 MGEMOMV     1.0
 MGEMLEEF    1.0
 MOSHOOFD    1.0
            …
 AZEILPL     1.0
 APLEZIER    1.0
 AFIETS      1.0
 AINBOED     1.0
 ABYSTAND    1.0
 Length: 85, dtype: float64)
```

3. Split the datasets into a test set containing the first 1,000 observations and a training set, including the remaining observations.

[57]:
```
y_value = caravan["Purchase"]
X_test, X_train, y_test, y_train = X_scaled_df[:1000], X_scaled_df[1000:],␣
↪y_value[:1000], y_value[1000:]
```

(a) How many observations are in each set?

[58]:
```
test_observations = len(X_test)
train_observations = len(X_train)
(train_observations, test_observations)
```

[58]:
```
(4822, 1000)
```

There are 4822 observations in training set and 1000 observations in testing set.

(b) How many customers purchased insurance in each set?

[59]:
```
purchased_insurance_in_test = (y_test == "Yes").sum()
purchased_insurance_in_train = (y_train == "Yes").sum()
(purchased_insurance_in_train, purchased_insurance_in_test)
```

[59]:
```
(289, 59)
```

Therefore, 289 customers purchased in training set whereas 59 purchased in testing set.

4. Binary Classifier: KNN and SGD classifiers

(a) Apply the K-Nearest Neighbors (KNN) classifier to the caravan dataset. Choose the values K = 1, 3, and 5, and for each K, compute the precision and recall. Please comment on the precision.

(i) K = 1

[63]:
```
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

[28]:
```
precision = precision_score(y_test, y_pred, pos_label='Yes')
precision
```

[28]:
```
0.11688311688311688
```

[30]:
```
recall = recall_score(y_test, y_pred, pos_label='Yes')
recall
```

[30]:
```
0.15254237288135594
```

(ii) K = 3

[64]:
```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

[34]:
```
precision = precision_score(y_test, y_pred, pos_label='Yes')
precision
```

[34]:
```
0.2
```

[35]:
```
recall = recall_score(y_test, y_pred, pos_label='Yes')
recall
```

[35]:
```
0.0847457627118644
```

(iii) K = 5

[65]:
```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

[37]:
```
precision = precision_score(y_test, y_pred, pos_label='Yes')
precision
```

[37]: 0.26666666666666666

[38]:
```
recall = recall_score(y_test, y_pred, pos_label='Yes')
recall
```

[38]: 0.06779661016949153

[39]:
```
f1score = f1_score(y_test, y_pred, pos_label='Yes')
f1score
```

[39]: 0.10810810810810811

As we can see from the calculations above, the precision has increased when the value of k increases from 1 to 5.

(b) Next, apply the Stochastic Gradient Descent (SGD) classifier on the caravan dataset. Compute the precision and recall. Set random seed to 42.

[41]:
```
stochasticgd_clf = SGDClassifier(max_iter=1000, tol=1e-3, random_state=42)
stochasticgd_clf.fit(X_train, y_train)
y_pred = stochasticgd_clf.predict(X_test)
```

[42]:
```
precision = precision_score(y_test, y_pred, pos_label='Yes')
precision
```

[42]: 0.3125

[43]:
```
recall = recall_score(y_test, y_pred, pos_label='Yes')
recall
```

[43]: 0.0847457627118644

[44]:
```
f1score = f1_score(y_test, y_pred, pos_label='Yes')
f1score
```

[44]: 0.13333333333333333

(c) Which classifier finds real patterns in the caravan dataset? The model that has the highest F1 score is the best. The SFD classifier achieves a higher F1 score (0.1333) compared to the KNN classifier with k=5 (0.1081). This suggests that, based on the F1 score, the SFD classifier is a better choice and appears to identify meaningful patterns in the caravan dataset.

[ ]: