Bhandari_Nischal_HW4cmps320

November 13, 2023

1 Part A

1.0.1 1

If your model performs great on the training data but generalizes poorly to new instances, what is happening? Discuss two possible solutions

Ans If a model performs great on the training data but generalizes poorly to new instances, then it is **overfitting**. It means the model is trying to memorize the variability in the data rather than capturing it for general purposes.

Two possible solutions to address overfitting are:

1. Regularization: Regularization techniques such as L1 or L2 regularization can be applied to the model. Regularization adds a penalty term which discourages the model from assigning too much importance to any particular feature in the dataset.

2. Increasing Training Data: With more diverse and representative data, the model can learn a better representation of the underlying patterns and generalize well to new instances.

1.0.2 2

Suppose we want to compute 5-Fold Cross-Validation error on 200 training examples. We need to compute error N1 times, and the Cross-Validation error is the average of the errors. To compute each error, we need to build a model with data of size N2 and test the model on the data of size N3. What are the appropriate numbers for N1, N2, N3?

N1 = 5

N2 = 160 (40 is the size of each training fold and we have 4 fold)

N3 = 40

1.0.3 3

From the given matrix, TP = 970, FP = 10, TN = 15, FN = 25 a. **Accuracy** = (970 + 10)/ 1020 = 0.97 b. **Accuracy of a majority-class line** = (TP + FP) / Total number of instances = (970 + 15)/ 1020 = 0.95

c. In this case of unbalanced data, where the majority class is significantly more prevalent, accuracy alone may not be a reliable measure. While the classifier's accuracy of 97% is higher than the majority-class baseline's 95%, it is crucial to consider other evaluation metrics like precision, recall, and F1 score. In general, yes, the classifier has higher accuracy.

d. Precision = TP / TP + FP = 970.980 = 0.99

e. Recall = TP / TP + FN = 970 / 995 = 0.97

f. Recall = 2* precision * recall / precion + recall = 0.98

2 Part B

[1]:
```python
import pandas as pd
```

2.0.1 1

[2]:
```python
# loading the data
insurance_df = pd.read_csv("Caravan_Homework_4.csv")
```

[3]:
```python
insurance_df = insurance_df.iloc[:, 1:]
insurance_df.head()
```

[3]:
```
   MOSTYPE  MAANTHUI  MGEMOMV  MGEMLEEF  MOSHOOFD  MGODRK  MGODPR  MGODOV  \
0       33         1        3         2         8       0       5       1
1       37         1        2         2         8       1       4       1
2       37         1        2         2         8       0       4       2
3        9         1        3         3         3       2       3       2
4       40         1        4         2        10       1       4       1

   MGODGE  MRELGE  …  APERSONG  AGEZONG  AWAOREG  ABRAND  AZEILPL  APLEZIER  \
0       3       7  …         0        0        0       1        0         0
1       4       6  …         0        0        0       1        0         0
2       4       3  …         0        0        0       1        0         0
3       4       5  …         0        0        0       1        0         0
4       4       7  …         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 rows x 86 columns]
```

[4]:
```python
# a
print("The dimension of the dataset is:", insurance_df.shape)
```

```
The dimension of the dataset is: (5822, 86)
```

[5]:
```python
# b
```

```python
demographic_predictors = sum(col.startswith('M') for col in insurance_df.
    ↪columns)
print("The number of predictors measuring demographic characteristics is:",␣
    ↪demographic_predictors)
```

```
The number of predictors measuring demographic characteristics is: 43
```

[6]:
```python
# c
caravan_purchased = insurance_df['Purchase'].value_counts(normalize=True)[1] *␣
    ↪100
print("The percentage of people who purchased caravan insurance is:",␣
    ↪round(caravan_purchased, 2), "%")
```

```
The percentage of people who purchased caravan insurance is: 5.98 %
```

2.0.2 2

Data processing

[7]:
```python
from sklearn.preprocessing import scale

# Standardizing the data such that all variables have a mean of 0 and sd of 1
insurance_df.iloc[:, :-1] = scale(insurance_df.iloc[:, :-1], axis=0)
insurance_df.head()
```

[7]:
```
    MOSTYPE  MAANTHUI   MGEMOMV  MGEMLEEF  MOSHOOFD    MGODRK    MGODPR  \
0  0.680906  -0.27258  0.406697 -1.216964  0.779405 -0.694311  0.217444
1  0.992297  -0.27258 -0.859500 -1.216964  0.779405  0.302552 -0.365410
2  0.992297  -0.27258 -0.859500 -1.216964  0.779405 -0.694311 -0.365410
3 -1.187437  -0.27258  0.406697  0.010755 -0.970980  1.299414 -0.948264
4  1.225840  -0.27258  1.672893 -1.216964  1.479559  0.302552 -0.365410

     MGODOV    MGODGE    MRELGE  …  APERSONG   AGEZONG  AWAOREG    ABRAND  \
0 -0.068711 -0.161816  0.427670  … -0.073165 -0.081055 -0.05992  0.764971
1 -0.068711  0.464159 -0.096077  … -0.073165 -0.081055 -0.05992  0.764971
2  0.914172  0.464159 -1.667319  … -0.073165 -0.081055 -0.05992  0.764971
3  0.914172  0.464159 -0.619824  … -0.073165 -0.081055 -0.05992  0.764971
4 -0.068711  0.464159  0.427670  … -0.073165 -0.081055 -0.05992  0.764971

    AZEILPL  APLEZIER   AFIETS   AINBOED  ABYSTAND  Purchase
0 -0.022706  -0.07365 -0.15062 -0.087348 -0.118816        No
1 -0.022706  -0.07365 -0.15062 -0.087348 -0.118816        No
2 -0.022706  -0.07365 -0.15062 -0.087348 -0.118816        No
3 -0.022706  -0.07365 -0.15062 -0.087348 -0.118816        No
4 -0.022706  -0.07365 -0.15062 -0.087348 -0.118816        No

[5 rows x 86 columns]
```

2.0.3 3

Splitting

[8]:
```
from sklearn.model_selection import train_test_split

# Splitting the datasets into a test set containing the first 1,000␣
↪observations and a training set
X = insurance_df.iloc[:, :-1]
y = insurance_df["Purchase"]
X_train, X_test, y_train, y_test = X[1000:], X[:1000], y[1000:], y[:1000]

#a Printing the dimensions of the training and test sets
print("The dimension of the training set is:", X_train.shape)
print("The dimension of the test set is:", X_test.shape)
```

```
The dimension of the training set is: (4822, 85)
The dimension of the test set is: (1000, 85)
```

[9]:
```
# b
train_purchased = y_train.value_counts()
test_purchased = y_test.value_counts()
print("Number of customers who purchased insurance in the training set:",␣
↪train_purchased[1])
print("Number of customers who purchased insurance in the test set:",␣
↪test_purchased[1])
```

```
Number of customers who purchased insurance in the training set: 289
Number of customers who purchased insurance in the test set: 59
```

2.0.4 4

Binary Classifier: KNN and SGD classifiers

[10]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import precision_score, recall_score

import warnings
warnings.filterwarnings("ignore", category=FutureWarning)


# Create KNN classifier with K = 1
knn_1 = KNeighborsClassifier(n_neighbors=1)
knn_1.fit(X_train, y_train)
y_pred_1 = knn_1.predict(X_test)
precision_1 = precision_score(y_test, y_pred_1, pos_label='Yes')
recall_1 = recall_score(y_test, y_pred_1, pos_label='Yes')

# Create KNN classifier with K = 3
```

```python
knn_3 = KNeighborsClassifier(n_neighbors=3)
knn_3.fit(X_train, y_train)
y_pred_3 = knn_3.predict(X_test)
precision_3 = precision_score(y_test, y_pred_3, pos_label='Yes')
recall_3 = recall_score(y_test, y_pred_3, pos_label='Yes')

# Create KNN classifier with K = 5
knn_5 = KNeighborsClassifier(n_neighbors=5)
knn_5.fit(X_train, y_train)
y_pred_5 = knn_5.predict(X_test)
precision_5 = precision_score(y_test, y_pred_5, pos_label='Yes')
recall_5 = recall_score(y_test, y_pred_5, pos_label='Yes')

# Print precision and recall for each K value
print("K = 1: Precision =", precision_1, "Recall =", recall_1)
print("K = 3: Precision =", precision_3, "Recall =", recall_3)
print("K = 5: Precision =", precision_5, "Recall =", recall_5)
```

```
K = 1: Precision = 0.11688311688311688 Recall = 0.15254237288135594
K = 3: Precision = 0.2 Recall = 0.0847457627118644
K = 5: Precision = 0.26666666666666666 Recall = 0.06779661016949153
```

As we jump from k = 1 to k = 5, the precision increases from 11 % to 26 %. A high precision indicates that the classifier is making accurate predictions of positive instances, while a low precision indicates a higher rate of false positives. So KNN performs best here when k = 5.

[11]:
```python
# Using SGD
from sklearn.linear_model import SGDClassifier


# Create SGD classifier
sgd = SGDClassifier(random_state= 42)
sgd.fit(X_train, y_train)
y_pred_sgd = sgd.predict(X_test)
precision_sgd = precision_score(y_test, y_pred_sgd, pos_label='Yes')
recall_sgd = recall_score(y_test, y_pred_sgd, pos_label='Yes')

# Print precision and recall for SGD classifier
print("SGD Classifier: Precision =", precision_sgd, "Recall =", recall_sgd)
```

```
SGD Classifier: Precision = 0.3125 Recall = 0.0847457627118644
```

[12]:
```python
from sklearn.metrics import f1_score

# k =5 because it was the best
f1_score_knn = f1_score(y_test, y_pred_5, pos_label='Yes')

# SGD
```

```python
f1_score_sgd = f1_score(y_test, y_pred_sgd, pos_label='Yes')

print("KNN Classifier: F1 Score =", f1_score_knn)
print("SGD Classifier: F1 Score =", f1_score_sgd)
```

```
KNN Classifier: F1 Score = 0.10810810810810811
SGD Classifier: F1 Score = 0.13333333333333333
```

The classifier that finds real patterns in the caravan dataset is the SGD classifier. This classifier has the highest F1 score, indicating a good balance between precision and recall. Therefore, it is the best model for this dataset.