CM5

February 25, 2021

1 [CM5] Covid Dataset (Naive Bayes)

1.1 Data Pre-processing

1.1.1 Importing Libraries

[1]:
```
import pandas as pd
from sklearn import preprocessing
from sklearn.model_selection import KFold,GridSearchCV,train_test_split
from sklearn.metrics import accuracy_score
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import CategoricalNB

import warnings
warnings.filterwarnings("ignore")
```

1.1.2 Loading dataset

[2]:
```
df_covid = pd.read_csv('covid_train.csv')
```

[3]:
```
df_covid.isnull().sum()
df_covid = df_covid.dropna(subset=['Age_Group'])
df_covid[['Outbreak_Related']] = df_covid[['Outbreak_Related']].
 ↪fillna(value="No")
```

Feature ‘age_group’ has 6 rows with missing values, we dropped those 6 rows as the data is large and there won’t be any data loss due to it. Also, Replaced the “NAN” values in feature ‘outbreak_related’ with “No” as only outbreak related cases are marked “Yes”.

1.1.3 One Hot Encoding

[4]:
```
# Chaning datatype of categorical variable from 'object' to 'category'
for col in␣
 ↪['Client_Gender','Case_AcquisitionInfo','Reporting_PHU_City','Outbreak_Related','Outcome1']
 ↪
    df_covid[col] = df_covid[col].astype('category')

# One hot encoding
```

```
df_covid['Client_Gender'] = df_covid['Client_Gender'].cat.codes
df_covid['Case_AcquisitionInfo'] = df_covid['Case_AcquisitionInfo'].cat.codes
df_covid['Reporting_PHU_City'] = df_covid['Reporting_PHU_City'].cat.codes
df_covid['Outbreak_Related'] = df_covid['Outbreak_Related'].cat.codes

# Dividing dataframe
df_covid_lb = df_covid.copy()
df_covid_num = df_covid.copy()

labelencoder = preprocessing.LabelEncoder()

# Case-1 : Label encoding for age as it is not a categorical variable
df_covid_lb['Age_Group'] = labelencoder.fit_transform(df_covid_lb['Age_Group'])
df_covid_lb['Outcome1'] = labelencoder.fit_transform(df_covid_lb['Outcome1'])

# Case-2 : Changing age to numarical value
df_covid_num['Age_Group'] = df_covid_num['Age_Group'].apply(lambda x: x.
 ↪strip('s'))
df_covid_num['Age_Group'] = df_covid_num['Age_Group'].replace({"<20": "19"})
df_covid_num['Outcome1'] = labelencoder.fit_transform(df_covid_num['Outcome1'])
```

1.1.4 Seperating X and y (Without Standardization)

[5]:
```
# Case-1 : Label encoding for age as it is not a categorical variable
X1 = df_covid_lb.iloc[:, 0:7]
y1 = df_covid_lb.iloc[:,7]
X_train1, X_test1, y_train1, y_test1 = train_test_split(X1, y1, test_size=0.2,␣
 ↪random_state=0)

# Case-2 : Changing age to numarical value
X2 = df_covid_num.iloc[:, 0:7]
y2 = df_covid_num.iloc[:,7]
X_train2, X_test2, y_train2, y_test2 = train_test_split(X2, y2, test_size=0.2,␣
 ↪random_state=0)
```

1.1.5 Seperating X and y (With Standardization)

[6]:
```
scaler = StandardScaler()

# Case-1 : Label encoding for age as it is not a categorical variable
XX1 = df_covid_lb.iloc[:, 0:7]
yy1 = df_covid_lb.iloc[:,7]
XX1 = scaler.fit_transform(XX1)
XX_train1, XX_test1, yy_train1, yy_test1 = train_test_split(XX1, yy1,␣
 ↪test_size=0.2, random_state=0)

# Case-2 : Changing age to numarical value
```

```
XX2 = df_covid_num.iloc[:, 0:7]
yy2 = df_covid_num.iloc[:,7]
XX2 = scaler.fit_transform(XX2)
XX_train2, XX_test2, yy_train2, yy_test2 = train_test_split(XX2, yy2,␣
↪test_size=0.2, random_state=0)
```

1.2 Naive Bayes Classifier (Without Standardization)

1.2.1 Case-1 : Label encoding for age as it is not a categorical variable

[7]:
```
kf = KFold(random_state=0, n_splits=10)
param_grid={'var_smoothing':[1e-10, 1e-9, 1e-5, 1e-3, 1e-1]}

classifier = GridSearchCV(GaussianNB(), param_grid=param_grid, cv=kf,␣
↪scoring="accuracy", n_jobs=-1)
classifier = classifier.fit(X_train1, y_train1)

results = classifier.cv_results_
print(results['mean_test_score'])
print(results['rank_test_score'])
```

```
[0.562229   0.562229   0.56256599 0.56483971 0.54016705]
[3 3 2 1 5]
```

1.2.2 Case-2 : Changing age to numarical value

Applying algorithm on training set

[8]:
```
kf = KFold(random_state=0, n_splits=10)
param_grid={'var_smoothing':[1e-10, 1e-9, 1e-5, 1e-3, 1e-1]}

classifier = GridSearchCV(GaussianNB(), param_grid=param_grid, cv=kf,␣
↪scoring="accuracy", n_jobs=-1)
classifier = classifier.fit(X_train2, y_train2)

results = classifier.cv_results_
print(results['mean_test_score'])
print(results['rank_test_score'])
```

```
[0.5827746  0.5827746  0.58395341 0.59826906 0.59372147]
[4 4 3 1 2]
```

Applying algorithm on test set

[9]:
```
clf = classifier.best_estimator_
clf.fit(X_train2, y_train2)
predictions = clf.predict(X_test2)
print(accuracy_score(y_test2, predictions))
```

```
0.5988548332771977
```

1.3 Naive Bayes Classifier (With Standardization)

1.3.1 Case-1 : Label encoding for age as it is not a categorical variable

[10]:
```
kf = KFold(random_state=0, n_splits=10)
param_grid={'var_smoothing':[1e-10, 1e-9, 1e-5, 1e-3, 1e-1]}

classifier = GridSearchCV(GaussianNB(), param_grid=param_grid, cv=kf,␣
 ↪scoring="accuracy", n_jobs=-1)
classifier = classifier.fit(XX_train1, yy_train1)

results = classifier.cv_results_
print(results['mean_test_score'])
print(results['rank_test_score'])
```

```
[0.562229   0.562229   0.562229   0.56206058 0.55650346]
[1 1 1 4 5]
```

1.3.2 Case-2 : Changing age to numarical value

Applying algorithm on training set

[11]:
```
kf = KFold(random_state=0, n_splits=10)
param_grid={'var_smoothing':[1e-10, 1e-9, 1e-5, 1e-3, 1e-1]}

classifier = GridSearchCV(GaussianNB(), param_grid=param_grid, cv=kf,␣
 ↪scoring="accuracy", n_jobs=-1)
classifier = classifier.fit(XX_train2, yy_train2)

results = classifier.cv_results_
print(results['mean_test_score'])
print(results['rank_test_score'])
```

```
[0.5827746  0.5827746  0.5827746  0.5827746  0.58277432]
[1 1 1 1 5]
```

Applying algorithm on test set

[12]:
```
clf = classifier.best_estimator_
clf.fit(XX_train2, yy_train2)
predictions = clf.predict(XX_test2)
print(accuracy_score(yy_test2, predictions))
```

```
0.5860559110811722
```

1.4 Observation

The highest accuracy achieved using Gaussian NB algorithm in 59.82. For final testing, we have selected the Case-2, where 's' were removed and '<20' replaced with '19' from feature 'age_group'. Whereas, in Case-1 feature 'age_group' is label encoded, and it has lower accuracy compared to Case-2. In final test set, we got 59.88 accuracy.

The accuracy varies with varying var_smoothing parameter in Gaussian NB. It can be observed in Case-2, as the value of var_smoothing parameter increases, the mean_test_score of the algorithm increases as well. But, again for too high var_smoothing value the accuracy of algorithm starts decreasing. The highest accuracy achieved, which is 59.82, is at var_smoothing value of 1e-3.

As small value of var_smoothing might miss the cases with higher variance, while too big var_smoothing values might take in consideration values with least correlations, so the best fit for algorithm is 1e-3(neither small nor big).

With standardization, accuracy decreases from 59.88 to 58.60

1.5 References

https://scikit-learn.org/stable/modules/preprocessing.html
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html
https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html