XGBOOST

Stanley Perez, Yiqing Guo, Dean Papadopoulos, Seungmin Kim

Overview

- Dataset
- Data Processing
- Model Implementation
- Model Optimization

Dataset Used

- The dataset used looks to see why 30% of patients miss their scheduled appointments.
- With over 110,000 records and 14 variables this dataset looks at Brazilian healthcare data.
- For this model we decided to analyze the Hipertension variable.
- Source: Medical Appointment No Shows | Kaggle

Data Processing

```
#Run correlation matrix to see which columns are relevant
df.corr()
```

	PatientId	AppointmentID	Age	Scholarship	Hipertension	Diabetes	Alcoholism	Handcap	SMS_received
PatientId	1.000000	0.004039	-0.004139	-0.002880	-0.006441	0.001605	0.011011	-0.007916	-0.009749
AppointmentID	0.004039	1.000000	-0.019126	0.022615	0.012752	0.022628	0.032944	0.014106	-0.256618
Age	-0.004139	-0.019126	1.000000	-0.092457	0.504586	0.292391	0.095811	0.078033	0.012643
Scholarship	-0.002880	0.022615	-0.092457	1.000000	-0.019729	-0.024894	0.035022	-0.008586	0.001194
Hipertension	-0.006441	0.012752	0.504586	-0.019729	1.000000	0.433086	0.087971	0.080083	-0.006267
Diabetes	0.001605	0.022628	0.292391	-0.024894	0.433086	1.000000	0.018474	0.057530	-0.014550
Alcoholism	0.011011	0.032944	0.095811	0.035022	0.087971	0.018474	1.000000	0.004648	-0.026147
Handcap	-0.007916	0.014106	0.078033	-0.008586	0.080083	0.057530	0.004648	1.000000	-0.024161
SMS_received	-0.009749	-0.256618	0.012643	0.001194	-0.006267	-0.014550	-0.026147	-0.024161	1.000000

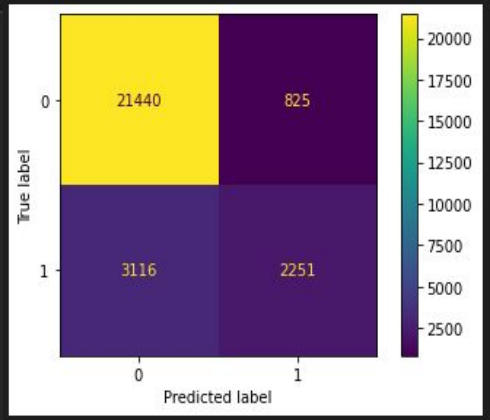
```
# create dummy variable for male and female
df = pd.get_dummies(df, columns= ['Gender'])
```

```
#Target Hipertension, drop columns unnecessary columns
X = df.copy()
X.drop(columns = ['PatientId','AppointmentID','ScheduledDay','AppointmentDay','No-show','Scholarship','SMS_received',
'Neighbourhood','Hipertension'], axis = 1, inplace = True)
y = df['Hipertension'].copy()
```

- Run correlation matrix
- Dummy variables for gender column
- Drop columns that had negative correlation to target column (Hypertension) & were non numeric

First run through with XGBoost

```
#Run our colleague's model

X_train, X_test, y_train, y_test = train_test_split( X, y, test_size = 0.25,random_state = 42 )

clf = xgb.XGBClassifier(objective='binary:logistic', seed = 0,early_stopping_rounds =10,eval_metric='aucpr')
clf.fit(X_train, y_train, eval_set=[(X_test,y_test)])

print(f'The training dataset has {len(X_train)} records.')
print(f'The testing dataset has {len(X_test)} records.')

y_predicted = clf.predict(X_test)

cm = confusion_matrix(y_test,y_predicted)

cmd = ConfusionMatrixDisplay(confusion_matrix=cm)

cmd.plot()

plt.show()
```




```
print(matthews_corrcoef(y_test,y_predicted))
```

```
0.48092689684388445
```

Matthew's Coefficient Score - 0.4809268
Training Dataset - 82,895
Testing Dataset - 27,632

Grid Search

```
estimator = xgb.XGBClassifier(
    objective= 'binary:logistic',
    nthread=4,
    seed=42
)

parameters = {
    'max_depth': range (2, 10, 1),
    'n_estimators': range(60, 220, 40),
    'learning_rate': [0.1, 0.01, 0.05]
}

grid_search = GridSearchCV(
    estimator=estimator,
    param_grid=parameters,
    scoring = 'roc_auc',
    n_jobs = 10,
    cv = 10,
    verbose=True
)
```

```
grid_search.fit(X, y)
```

```
Fitting 10 folds for each of 96 candidates, totalling 960 fits
```

```
print(grid_search.best_estimator_)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.05, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=9, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=180,
              n_jobs=4, nthread=4, num_parallel_tree=1, predictor='auto',
              random_state=42, reg_alpha=0, ...)
```

Optimized Model Results

Training Data - 82,895
Testing - 27,632

First Model -
Matthew's Coefficient Score - 0.4809268

Model after implementing Grid Search -
Matthew's Coefficient Score - 0.484427