Tickets_DataScience

September 2, 2019

[1]:
```
import numpy as nm
import pandas as pd
import os

train_df=pd.read_csv('train.csv')
test_df=pd.read_csv('test.csv')
```

[2]:
```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 389 entries, 0 to 388
Data columns (total 25 columns):
Reopened?                          389 non-null int64
Priority                           389 non-null int64
Assignment group                   389 non-null int64
Weightage                          389 non-null int64
Highest Assignment group           389 non-null int64
Support Level(IE?)                 389 non-null int64
Automation Pickup Time (Min)       389 non-null int64
Technician Pickup Time (Min)       389 non-null int64
IE Escalation Time (Hrs)           389 non-null float64
Business Resolution Time (Hrs)     389 non-null float64
Automation Pickup Time Range       389 non-null int64
Automation Pickup in < 5 Min?      389 non-null int64
IE Catgory                         389 non-null int64
technician Pickup in < 15 Min?     389 non-null int64
Week Resolved                      389 non-null int64
Week EscalaOd                      389 non-null int64
Resolved < 60 Min?                 389 non-null int64
Knowledge Article Used?            389 non-null int64
Hour Resolved                      389 non-null int64
KB Article Used?                   389 non-null int64
05 min or less?                    389 non-null int64
Cluster                            389 non-null int64
Initial Priority                   389 non-null int64
MTTR SLA Met?                      389 non-null int64
MTTA SLA Met?                      389 non-null int64
```

```
dtypes: float64(2), int64(23)
memory usage: 76.0 KB
```

[3]:
```
X=train_df.loc[:,'Priority':].as_matrix().astype('float')
y=train_df['Reopened?'].ravel()
```

```
/home/nbuser/anaconda2_501/lib/python2.7/site-packages/ipykernel/__main__.py:1:
FutureWarning: Method .as_matrix will be removed in a future version. Use
.values instead.
  if __name__ == '__main__':
```

[4]:
```
print X.shape, y.shape
```

```
(389, 24) (389,)
```

[5]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test=train_test_split(X,y,test_size=0.
 ↪2,random_state=0)
```

[6]:
```
print X_train.shape, y_train.shape

print X_test.shape, y_test.shape

print 'Mean Reopened in train : {0:.3f}'.format(nm.mean(y_train))

print 'Mean Reopened in test : {0:.3f}'.format(nm.mean(y_test))
```

```
(311, 24) (311,)
(78, 24) (78,)
Mean Reopened in train : 0.309
Mean Reopened in test : 0.308
```

[7]:
```
# we can see that reopened mean is same for both, so the dataset is splitted␣
 ↪properly
```

0.1 BaseLine Model

[8]:
```
from sklearn.dummy import DummyClassifier

model_dummy= DummyClassifier(strategy='most_frequent', random_state=0)

model_dummy.fit(X_train,y_train)

#giving inputs to the dummy algo or the baseline algo
```

[8]:
```
DummyClassifier(constant=None, random_state=0, strategy='most_frequent')
```

[9]:
```
print 'Score for baseline model : {0: .2f}'.format(model_dummy.score(X_test,␣
 ↪y_test))

# in this the model will first get a prediction as output from dummy model or␣
 ↪baseline model
# all these outputs will then be compared to the y_test which the actual result,␣
 ↪hence will give as an idea as to how accurately the algo is predicting by␣
 ↪comparing the predicted o/p to the actual values

#as the answer is 0.69, this means that 69% of times the baseline models␣
 ↪predicts right

# so without using ML, we are still getting .69 accuracy just by classification␣
 ↪for predicitng wheather a ticket will be reopened or not
```

```
Score for baseline model :  0.69
```

[10]:
```
import pandas
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

# prepare configuration for cross validation test harness
seed = 7
# prepare models
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
# evaluate each model in turn
results = []
names = []
scoring = 'accuracy'
for name, model in models:
        kfold = model_selection.KFold(n_splits=10, random_state=seed)
        cv_results = model_selection.cross_val_score(model, X, y, cv=kfold,␣
 ↪scoring=scoring)
        results.append(cv_results)
        names.append(name)
```

```
        msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
        print(msg)
# boxplot algorithm comparison
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

```
/home/nbuser/anaconda2_501/lib/python2.7/site-
packages/matplotlib/font_manager.py:281: UserWarning: Matplotlib is building the
font cache using fc-list. This may take a moment.
  'Matplotlib is building the font cache using fc-list. '

LR: 0.766127 (0.123630)

/home/nbuser/anaconda2_501/lib/python2.7/site-
packages/sklearn/discriminant_analysis.py:388: UserWarning: Variables are
collinear.
  warnings.warn("Variables are collinear.")

LDA: 0.760999 (0.150736)
KNN: 0.735358 (0.092866)
CART: 0.758435 (0.101225)
NB: 0.686707 (0.226414)
SVM: 0.750742 (0.192124)

<Figure size 640x480 with 1 Axes>
```

[ ]:
```
# so we can predict upto 76.6% using Logistic Regression and 76% with␣
↪LinearDiscriminantAnalysis that whether a ticket will reopen or not. This is␣
↪higher than baseline model of 69%.
```