ACME Happiness Survey Analysis

Overview

Client is one of the fastest growing startups in the logistics and delivery domain. They work with several partners and make on-demand delivery to their customers. With a global expansion strategy, they need to make their customers happy and therefore, they want to measure how happy each customer is. If they can predict what makes our customers happy or unhappy, they can take necessary actions.

Data and model

Data

Client recently did a survey to a select customer cohort. Because it’s not easy to collect data from customers, our data is quite small with only 126 entries. Below is data description:

Y = target attribute (Y) with values indicating 0 (unhappy) and 1 (happy) customers
X1 = my order was delivered on time
X2 = contents of my order was as I expected
X3 = I ordered everything I wanted to order
X4 = I paid a good price for my order
X5 = I am satisfied with my courier
X6 = the app makes ordering easy for me

Attributes X1 to X6 indicate the responses for each question and have values from 1 to 5 where the smaller number indicates less and the higher number indicates more towards the answer.

Preparing data

There are few challenges with this problem, particular with the given data set. The data set is quite small and imbalanced - there are more happy customers than unhappy ones. Even though this is good for our client, we need more information on the unhappy customers to achieve better prediction. Furthermore, the client is more interested in what makes a customer unhappy so that they can take actions to prevent it.

SMOTE (Synthetic Minority Oversampling techniques) is used to increase data points of the underrepresented class - unhappy customers.

```
X = df.drop(columns='Y')
y = df.Y
oversample = SMOTE(random_state=seed)
X, y = oversample.fit_resample(X, y)
```

RFE (Recursive Feature Elimination) is used to eliminate features that don’t contribute positively to model prediction

```
# define the method
rfe = RFE(estimator=DecisionTreeClassifier(random_state=seed),
n_features_to_select=4)

# fit the model
rfe.fit(X_train, y_train)

# transform the data
X_train_rfe = rfe.transform(X_train)
X_test_rfe = rfe.transform(X_test)
```

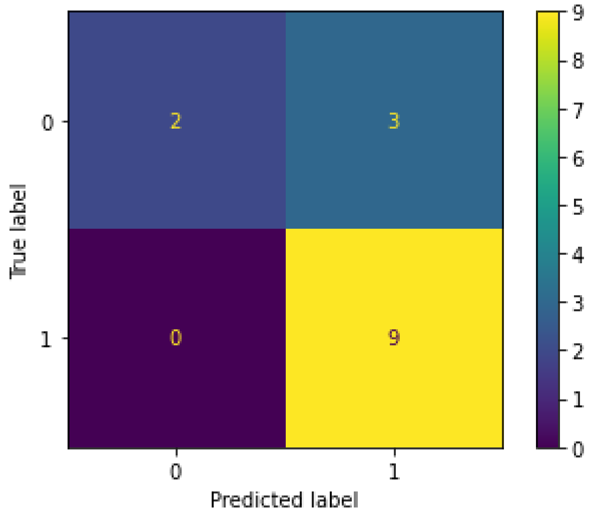
The resulting chosen features are X1, X2, X3, X5 while X4 and X6 are eliminated. This suggests that price and ease of use of is less important to customers comparing to having their orders delivered on time, as expected and completed

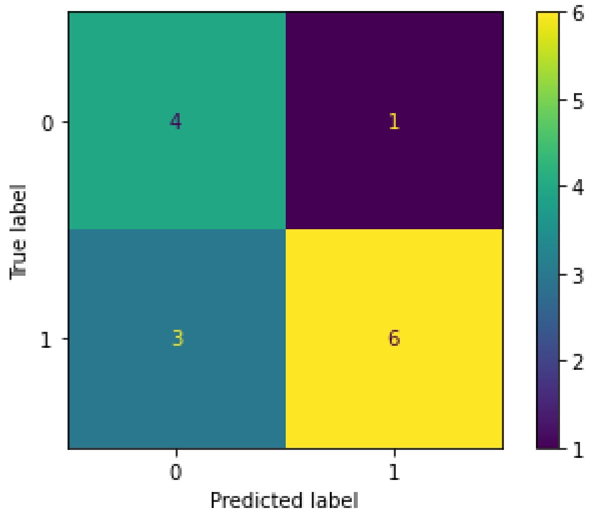
Model and Result

For initial assessment of models, Lazy Classifier is used to quickly see and get an idea of which models would be best to try and potentially improve upon

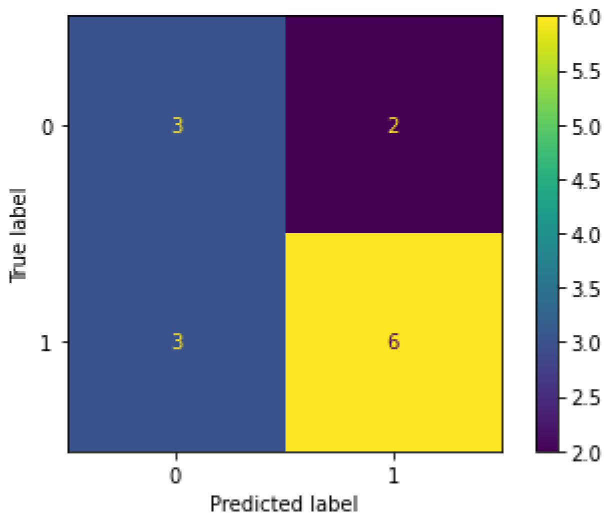
```
clf = LazyClassifier(verbose=0,ignore_warnings=True, custom_metric=None,
random_state=seed)
models,predictions = clf.fit(X_train_rfe, X_test_rfe, y_train, y_test)

print(models)
```

Based on the suggested models, the following models are used

1. Passive Aggressive Classifier:
```
Score:  0.5403225806451613
CV average score: 0.56
```


2. BernoulliNB
```
Score:  0.6774193548387096
CV average score: 0.63
```


3. Ridge Classifier CV
```
Score:  0.6370967741935484
CV average score: 0.61
```



In an effort improve model's accuracy score, ensemble methods are tested:

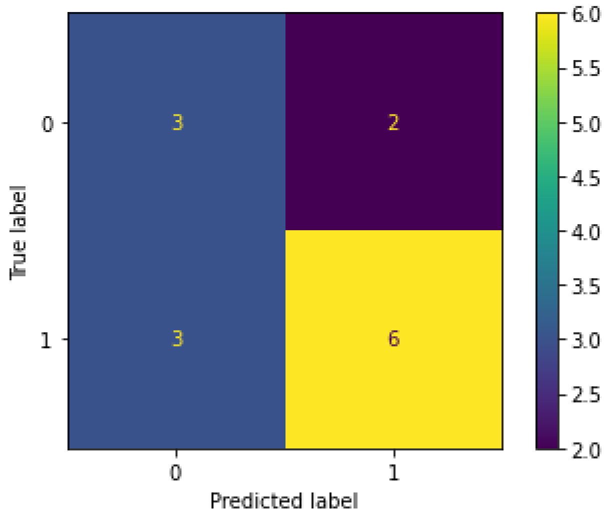
1. Majority Voting
   The three models used in majority voting are: Passive Aggressive, BernoulliNB, Ridge Classifier CV

```
clf1 =
make_pipeline(SimpleImputer(),StandardScaler(),PassiveAggressiveClassifier(rand
om_state=seed))
clf2 = make_pipeline(SimpleImputer(),StandardScaler(),BernoulliNB())
clf3 = make_pipeline(SimpleImputer(),StandardScaler(),RidgeClassifierCV())
```

```
eclf1 = VotingClassifier(estimators=[('PasAgg', clf1), ('BNB', clf2), ('Ridge',
clf3)], voting='hard')
eclf1 = eclf1.fit(X_train_rfe, y_train)
# evaluate the model
scores = evaluate_model(eclf1, X_train_rfe, y_train)
print("Score: ", score)
cv_scores = cross_val_score(eclf1, X_train_rfe, y_train, cv=3 )
print("CV average score: %.2f" % cv_scores.mean())
```

The result from this method isn't as promising as hoped:

```
Score:  0.6370967741935484
CV average score: 0.62
```

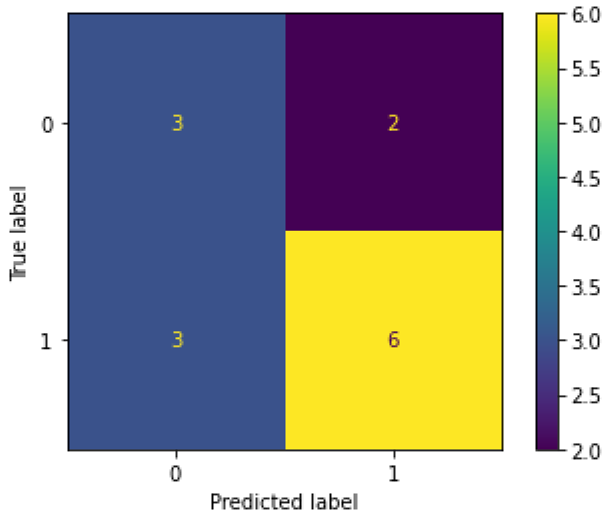


2. Stacking Classifier
   For stacking classifier, Passive Aggressive, BernoulliNB and Logistic Regression as final estimator

```
estimators = [('PasAgg', clf1),('BNB', clf2)]
clf = StackingClassifier(
   estimators=estimators,
final_estimator=LogisticRegression(random_state=seed))
clf.fit(X_train_rfe, y_train).score(X_test_rfe, y_test)
```

Result for this method is more promising:

```
Score:  0.6370967741935484
CV average score: 0.68
```



Conclusion

It’s quite challenging to predict if a customer is happy or not. It seems that customer’s satisfaction correlated much better with the quality of order and the service in comparison with price and ease of use. The reason for this could be that customers who placed orders are informed of the price and accept it ahead of time. It’s possible that potential customers who are not accepting of the price didn’t order and therefore are not part of this survey. Similarly, potential customers who have trouble navigating the app have a higher chance of not finishing their order, leading to this survey only collecting data from people who don’t mind the app interface. While the initial correlation makes sense, the predictive power of many models isn’t great. It might be beneficial to look into existing bias within the data set. Otherwise, for people who already placed their orders with the client, the stacking assemble method described above can predict whether a customer is happy or not with approximately 68% accuracy.