Name: Tan Nguyen

CS 422

INTRODUCTION TO MACHINE LEAERNING

FALL 2023

ASSIGNMENT 3

Machine Learning Report: Credit Card Approval Prediction Using Logistic Regression and Naïve Bayes Models

Dataset Source:

The dataset from Kaggle, titled "Credit Card Approval," consists of credit card application data submitted to a commercial bank. It's a classic dataset used in machine learning to build classification models that predict whether an application will be approved or denied based on various attributes.

Field name	Description	Type
Gender	Customer's gender	Cat
Age	Customer's age as of cut off date	Float
Debt	Amount of debt balance	Float
Married	Marital status	Cat
BankCustomer	Customer category	Cat
EducationLevel	Customer's education category	Cat
Ethnicity	Customer's ethnicity	Cat
YearsEmployed	Customer's years of employment	Float
PriorDefault	If default before	Bool
Employed	If currently employed	Bool
CreditScore	Customer's credit score	Int
DriverLicense	If customer has driving license	Bool
Citizen	Customer's citizenship	Cat
ZipCode	Primary's zip code	String
Income	Monthly income	Int
ApprovalStatus	If approved for credit card	Bool

Data Processing:

- Transform data set into pd.DataFrame
- As provided dataset is missing column headers, need to add in
- Encode all string-type features using `LabelEncoder()`
- The dataset was split into an 80/20 ratio for training and testing, respectively.

Parameter Vector:

```
log_model = SGDClassifier(loss='log_loss', max_iter=500, tol=1e-2,
                          random_state=42, alpha=1, penalty='elasticnet', l1_ratio=0.2,
                          early_stopping=True, validation_fraction=0.1
                          , learning_rate='adaptive', eta0=0.012)
```

Tunning the SGDClassifier is a bit tricky, but it is found that assigning 'alpha' – the regularization term has the most impact on the performance.

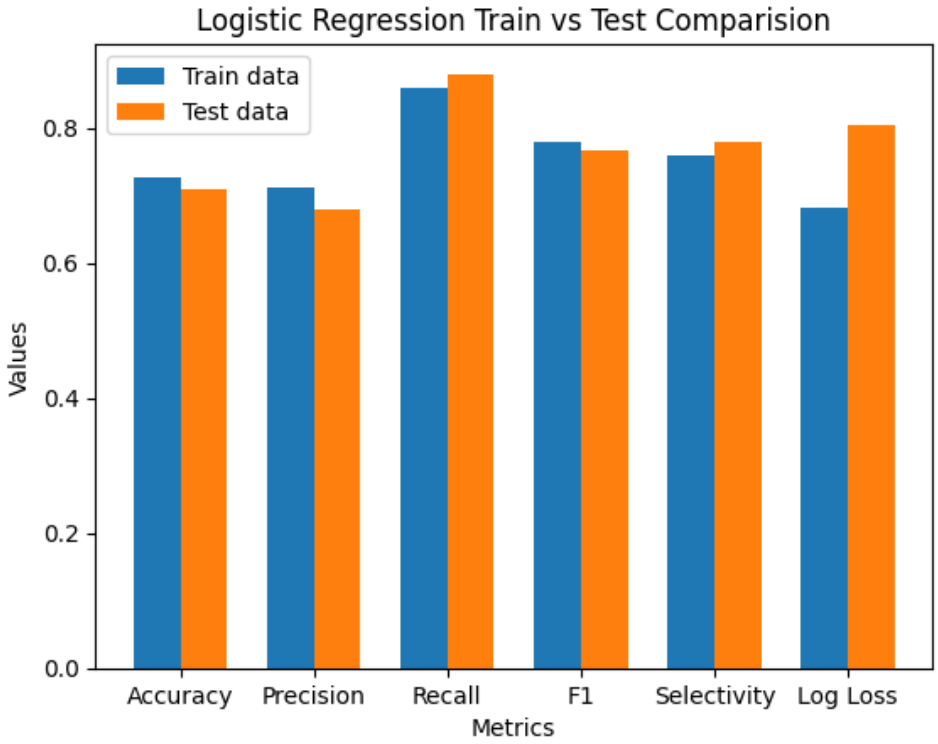
- 'alpha' < 1 : less penalty -> overfitting -> high log_loss
- 'alpha' > 1 : all logistic parameters in w turn near 0
- 'alpha' = 1 : so far provide the best performance, on par with the 'sklearn.LogisticRegression()'

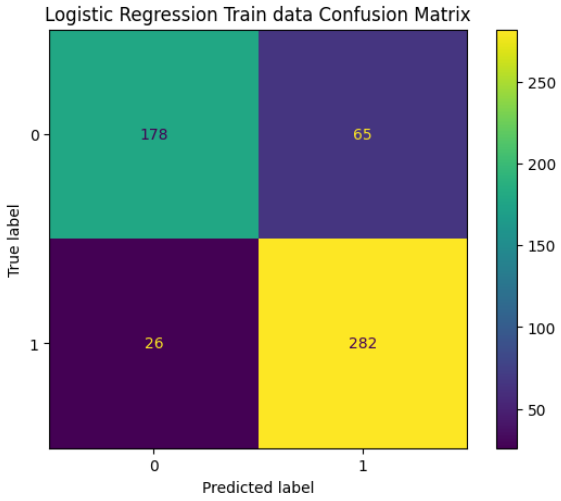
```
w0 = log_model.intercept_
w1 = log_model.coef_[0]

w = np.concatenate((w0.reshape(1,1), w1.reshape(1,15)), axis=1).flatten()
print('Logistic Regression parameter vector:', ' '.join([f'{val:.2f}' for val in w]))
```

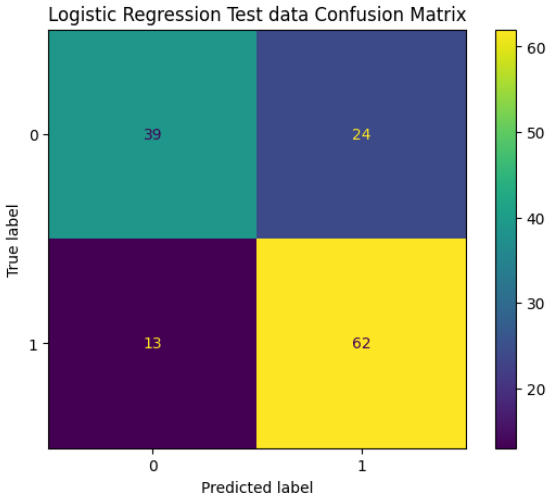
```
W = [ 3.20 0.00 -0.01 0.00 0.00 0.00 0.00 0.00 -0.17 0.00 0.00 -0.14 0.00 0.00
0.00 -0.00]
```

Intercept: **3.20**

Evaluation – Train vs Test data:

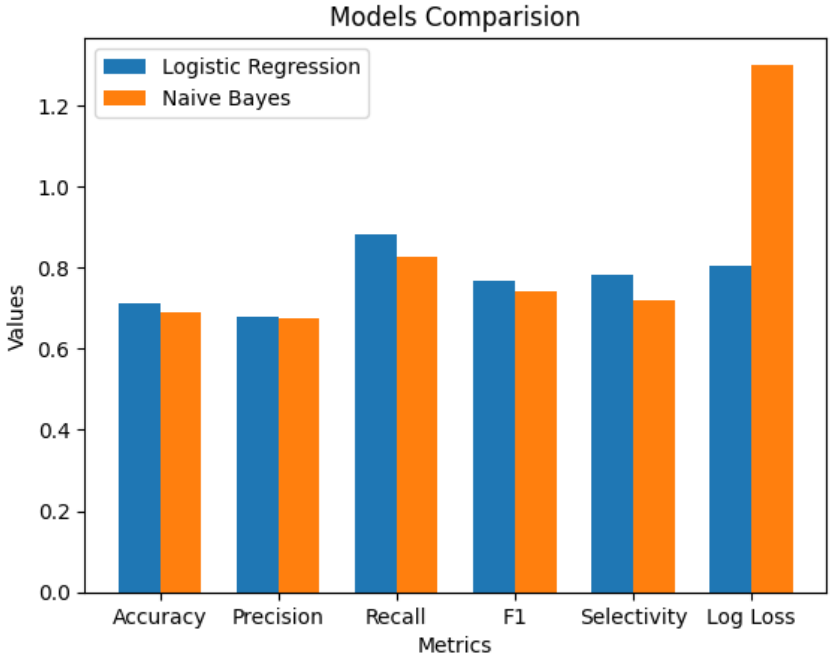


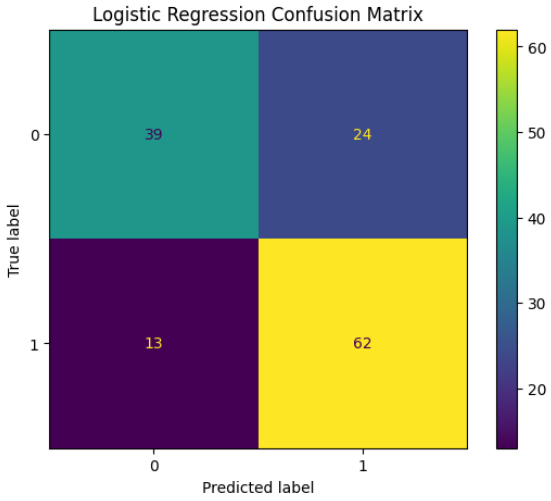


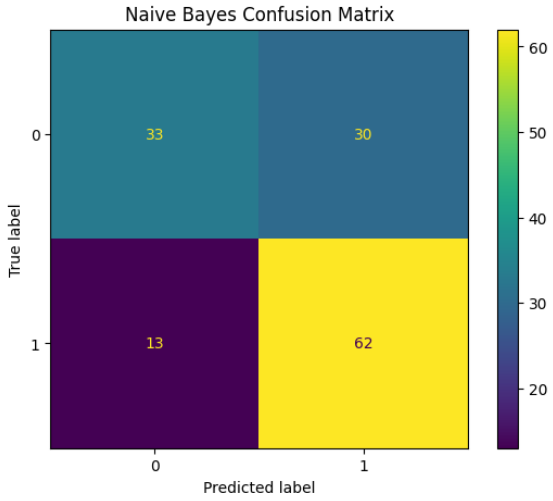


The Logistic Regression model demonstrates decent performance on both the training and testing datasets. However, it seems to be slightly overfitting the training data, as indicated by higher performance metrics for the training set compared to the test set. The false positive rate in the training data also reinforces this observation. Measures might be needed to mitigate this overfitting, such as regularization or further feature engineering.

Evaluation – Logistic Regression vs Naïve Bayes Gaussian:







Logistic regression slightly outperforms GaussianNB in most of the metrics, with more consistent error rate, as it has lower log loss. The potential explanation would be the underlying assumptions of each model. Logistic regression might be better suited for this dataset as it does not assume features to be conditionally independent within one class as GaussianNB does.