Predicting Conversion Rates for Optimal Budget Allocation

Liam Resnick - 11th Grade
Lower Moreland High School

Definitions

AUC score - Is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

Response Coding - It is a technique to represent the categorical data while solving a machine learning classification problem.

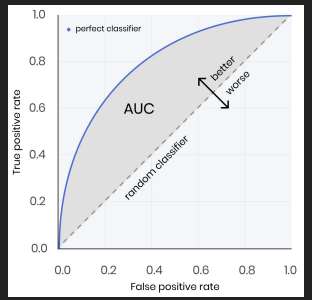
Logistic Regression - It is used for predicting the categorical dependent variable using a given set of independent variables.

T-SNE - Is a non-linear dimensionality reduction algorithm used for exploring high-dimensional data.

Conversion Rates - A conversion rate records the percentage of users who have completed a desired action.

Research

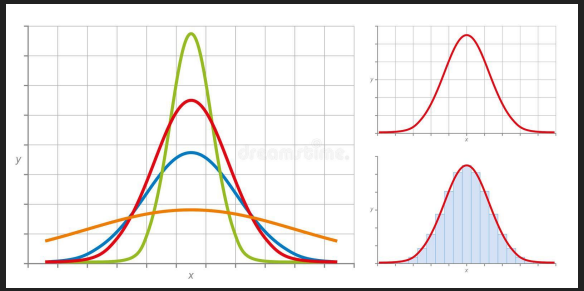
- The goal of this case-study is to determine if it's possible, to a degree of reasonable accuracy, to not only predict conversion rates, but to also apply that information in budget allocation
- Machine Learning: KNN, Linear SVM, Random Forest, XGBoost
  - Logistic Regression is a supervised learning classification algorithm used to predict the probability of a target variable.
  - It is also one of the simplest ML algorithms that can be used for various classification problems
- Using AUC over accuracy: Whereas using accuracy on any random model can result in a positive output, it will still be doing so regarding a random model, and AUC gets over this problem by looking into both the (TPR) and (FPR)



Source of Data

- The UCI Machine Learning Repository is a center for machine learning and intelligent-system data sets
  - In 2008-2010 the Bank Marketing Data Set was released according to research done by a Portuguese banking institution
- The data is related with direct marketing campaigns
- Users would answer 20 questions (input variables - x)
- Businesses and institutions can access these variables
- There were 4 data-sets provided , but the largest and most in depth was the *bank-additional-full* with all examples (41,188) and 20 inputs, ordered by date (from May 2008 to November 2010)

Background & Expected Outcome



- The work I did can be broken down into 3 parts:
  - Implementing a machine learning algorithm which can accurately identify the probability of subscription to a deposit based on user-inputs (this will be done using the 1000s of telephoned surveys),
  - And subsequently write a mathematical function for distributing a specified budget based off of the probability mentioned.
  - Finally, compare a standard, uniform distribution to the specified distribution function to quantify the improvement
- Ideally, the machine learning algorithm will accurately predict the likelihood of converting each individual which will allow me to optimize budget allocation more successfully than equal allocation for each customer.

Hypothesis & Null

- Hypothesis: Applying a logistic regression ML algorithm and likelihood-based budget allocation will produce a higher expected return than equal (uniform) allocation.
- Null: This method will not improve returns on the marketing budget beyond equal distribution.

Variables

Independent:

- The method of budget allocation

Dependent:

- Return on a given budget

Control:

- The data of consumers
- The pre established budget
- The Machine Learning algorithm used for the trials as well as for implementing visualization tools

Material list

Materials needed for the experiment:

- Laptop running windows with a core I7 9th generation processor
- Python 3
- Text Editor - VS Code
- Powershell
- Imports: matplotlib, pandas, sklearn

Procedure

1. Download the public “Bank Marketing Data Set” and work out the general structure of the machine learning algorithm
2. As a precursor to the Machine Learning section:
    a. Deal with missing values
    b. Deal with duplicate values
    c. Separate Independent and class variables
    d. Split the data
    e. Remove the duration column
3. Encode the subsets of data and apply T-SNE visualization
4. Implement the algorithm such that it takes the aforementioned data set and uses those values to create a reasonable conversion rate for any imputed values
5. Use the likelihood of conversion to distribute the budget in a sinusoidal fashion
6. Compare a standard distribution method to the new one, to quantify the improvement (if it exists).

Code

```
data_dup = data[data.duplicated(keep="last")]

#Prints (12, 21) indicating 12 rows of duplicates - each row with 21 columns
print(data_dup.shape)

#Drops duplicates
data = data.drop_duplicates()

#Will print out (41176, 21) indicating that it has dropped the 12 duplicates
print(data.shape)
```

```
data_x = data.iloc[:, :-1]
print("Shape of X:", data_x.shape)
data_y = data["y"]
print("Shape of Y:", data_y.shape)
```

```
# Remove duration feature
X_train = X_train.drop("duration", axis=1)
X_cv = X_cv.drop("duration", axis=1)
X_test = X_test.drop("duration", axis=1)

X_train.to_csv("Response_coded_features_train.csv")
X_cv.to_csv("Response_coded_features_cv.csv")
X_test.to_csv("Response_coded_features_test.csv")
```

```
from sklearn.model_selection import train_test_split

X_rest, X_test, y_rest, y_test = train_test_split(data_x, data_y, test_size=0.2)
X_train, X_cv, y_train, y_cv = train_test_split(X_rest, y_rest, test_size=0.2)

print("X Train:", X_train.shape)
print("X CV:", X_cv.shape)
print("X Test:", X_test.shape)
print("Y Train:", y_train.shape)
print("Y CV:", y_cv.shape)
print("Y Test:", y_test.shape)

y_train.replace({"no":0, "yes":1}, inplace=True)
y_cv.replace({"no":0, "yes":1}, inplace=True)
y_test.replace({"no":0, "yes":1}, inplace=True)

X_train.head()
```

Code

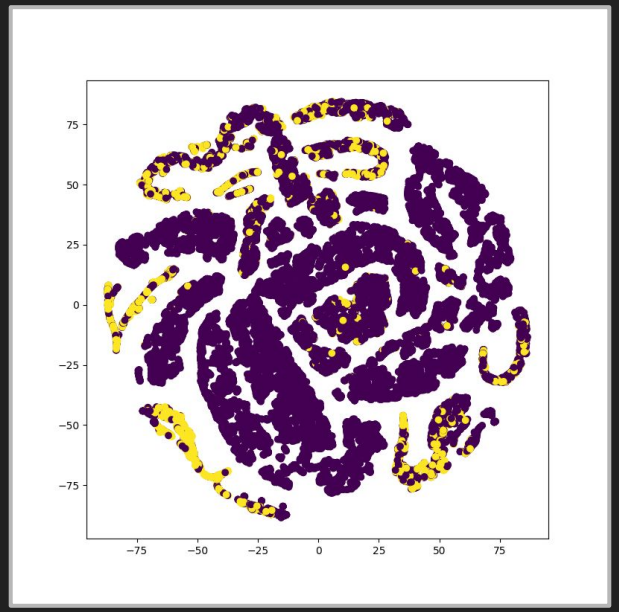
```
def get_fea_dict(alpha, feature, train_df, train_df_y):
    value_count = train_df[feature].value_counts()

    # Categorical feature Dict, which contains the probability array for each categorical variable
    feat_dict = dict()

    # denominator will contain the number of time that particular feature
    for i, denominator in value_count.items():

        vec = []
        for k in range(0, 2):

            cls_cnt = train_df.loc[(train_df_y==k) & (train_df[feature]==i)]
            vec.append((cls_cnt.shape[0] + alpha*10)/ (denominator + 20*alpha))

        feat_dict[i]=vec
    return feat_dict
```

```
def ResponseEncoder(categorical_cols, x_df, y_df):


    #This function takes Categorical column names and X and Y dataframe.
    #Returns the response coded dataframe

    print("Encoding Train dataset")
    print("Shape of the train dataset before encoding: ", X_train.shape)
    for i in (categorical_cols):
        temp_response_coded_feature = np.array(get_response_feature(alpha=1, feature=i, train_df=x_df, train_df_y=y_df))
    df_response = pd.DataFrame(temp_response_coded_feature, columns=[i+"_0", i+"_1"])
    x_df = pd.concat([x_df, df_response], axis=1)

    x_df = x_df.drop(categorical_cols, axis=1)
    return x_df
```

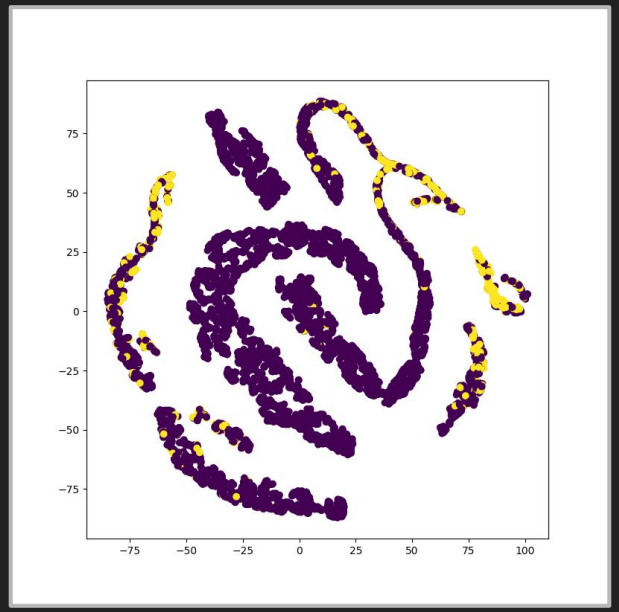
Code

```
alpha = [10 ** x for x in range(-5, 4)]
cv_auc_array=[]
for i in alpha:
    logisticR=LogisticRegression(penalty='l2',C=i,class_weight='balanced')
    logisticR.fit(X_train,y_train)
    sig_clf = CalibratedClassifierCV(logisticR, method="sigmoid")
    sig_clf.fit(X_train, y_train)
    predict_y = sig_clf.predict_proba(X_cv)
    cv_auc_array.append(roc_auc_score(y_cv, predict_y[:,1]))

for i in range(len(cv_auc_array)):
    print ('AUC for k = ',alpha[i],'is',cv_auc_array[i])

best_alpha = np.argmax(cv_auc_array)

fig, ax = plt.subplots()
ax.plot(alpha, cv_auc_array,c='g')
for i, txt in enumerate(np.round(cv_auc_array,3)):
    ax.annotate((alpha[i],np.round(txt,3)), (alpha[i],cv_auc_array[i]))
plt.grid()
plt.title("Cross Validation Error for each alpha")
plt.xlabel("Alpha i's")
plt.ylabel("Error measure")
plt.show()

logisticR=LogisticRegression(penalty='l2',C=alpha[best_alpha],class_weight='balanced')
logisticR.fit(X_train,y_train)
sig_clf = CalibratedClassifierCV(logisticR, method="sigmoid")
sig_clf.fit(X_train, y_train)
```

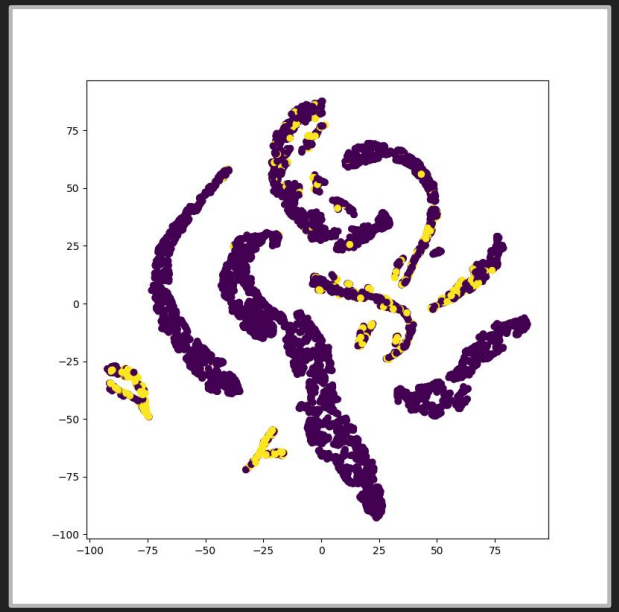
Visualization



T-SNE plot for train dataset

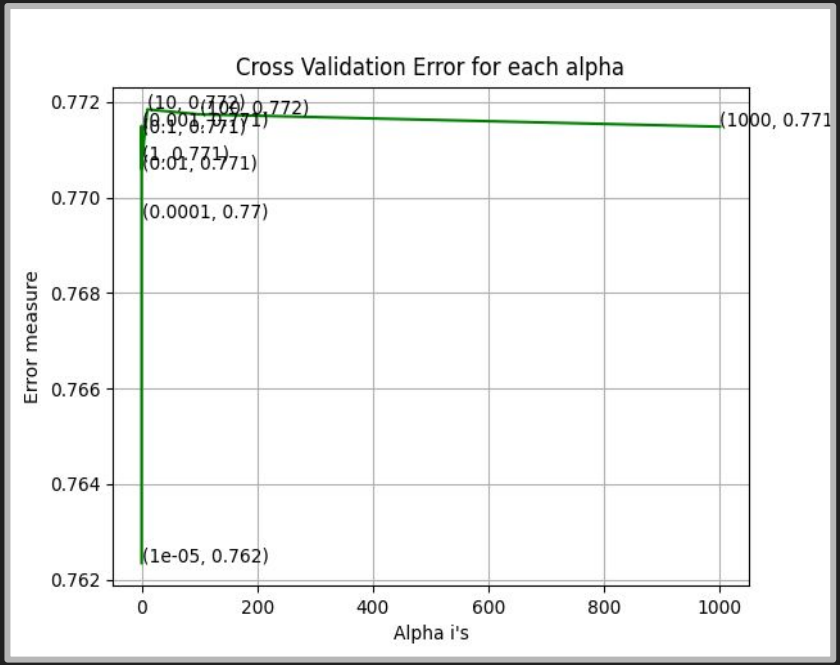


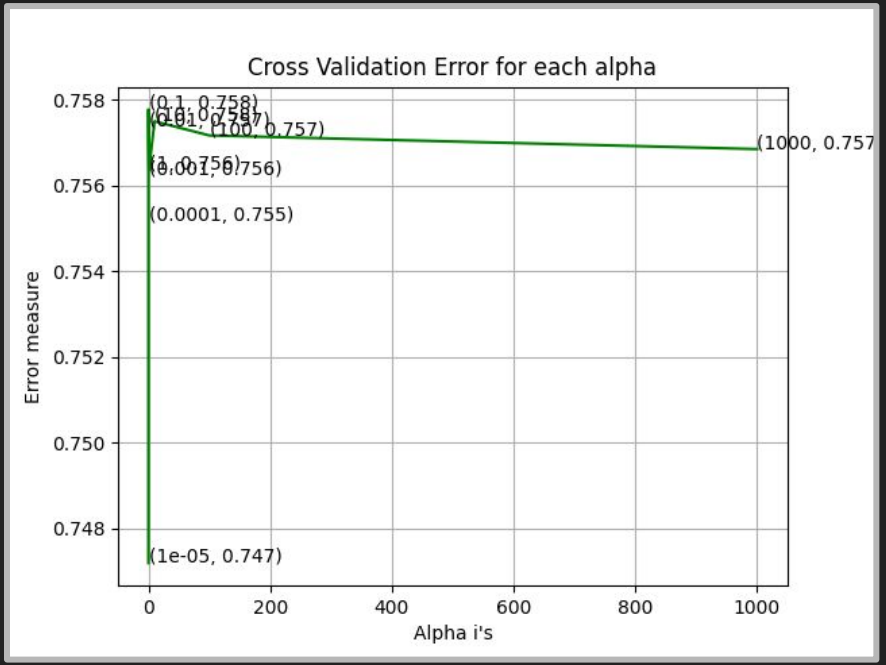
T-SNE plot for test dataset



T-SNE plot for CV dataset

Logistic Regression Results





For values of best alpha = 100 The train AUC is: 0.7932716928892307
For values of best alpha = 100 The cross validation AUC is: 0.7796918468051047
For values of best alpha = 100 The test AUC is: 0.7785456926327435

Part Two: Function

Tuesday, February 22, 2022 8:01 PM

for y in People:

Scalar = $\sin(P(y)\pi)$

Amount to spend for person i:

$$\left[\frac{\sin(P(y_i)\pi)}{\sum_{k=0}^{\infty}\sin(P(y_k)\pi)}\right] \cdot B$$

0.3, 0.5

↓ 0.8 ↓ 1

$\frac{0.8}{1.8} = 0.44$

$\frac{1}{1.8} = 0.55$

```python
budget = 10000 # The total budget for the list of people
l = [0.5, 0.7, 0.5, 0.3, 0.0023, 0.9919] # Output of PART 1 (likelihoods for each person)

def computeSinDist(x):
    return (math.sin(x * math.pi))

def computeAmountToSpend(budget, l):
    output = []
    total = 0

    # find total sum
    for user in l:
        total += computeSinDist(user)

    # compute each allocation amount
    for user in l:
        output.append((user, (computeSinDist(user)/total) * budget))

    return output

# prettify output
def prettyOutput(amnts):
    for amn in amnts:
        print(f'Likelihood: {amn[0]} | Allocation: $ {amn[1]}')

prettyOutput(computeAmountToSpend(budget, l))
```

```
PS C:\Users\liamr\OneDrive\Desktop\Unit 6 T\Studentsss> & C:/Users/liar
Likelihood: 0.5 | Allocation: $ 2739.197892613034
Likelihood: 0.7 | Allocation: $ 2216.057646079987
Likelihood: 0.5 | Allocation: $ 2739.197892613034
Likelihood: 0.3 | Allocation: $ 2216.057646079987
Likelihood: 0.0023 | Allocation: $ 19.79234891701636
Likelihood: 0.9919 | Allocation: $ 69.69657369694117
PS C:\Users\liamr\OneDrive\Desktop\Unit 6 T\Studentsss>
```

Part Three: Comparison

```python
def genProbs(numOfPeople):
    l = []
    for x in range(numOfPeople):
        l.append(random.random())
    return l
def computeUniSum(probs, budgetPerPerson):
    uniSum = 0
    for i in range(0, len(probs)):
        uniSum += budgetPerPerson * probs[i]
    return uniSum
def computeSmartSum(allocs):
    smartSum = 0
    for prob, alloc in allocs:
        smartSum += alloc * prob
    return smartSum
```

$$V_t = \sum_{i=0}^{n} P(x_i) \cdot B_i$$

```python
def main():
    numOfPeople = 1000
    budget = 10000
    # allocation for uniform distribution
    budgetPerPerson = budget / numOfPeople
    # probability-based allocation
    probs = genProbs(numOfPeople)
    allocs = computeAmountToSpend(budget, probs)
    # determine effectivness of probability-based allocation
    uniSum = computeUniSum(probs, budgetPerPerson)
    smartSum = computeSmartSum(allocs)
    print(f'Uniform: {uniSum} | Probability Based {smartSum}')
main()
```

```
Uniform:  5047.531929185237 | Probability Based 6160.648300900796
improvment: 1.2205268609158133
PS C:\Users\liamr\OneDrive\Desktop\PJAS 2022> & C:/Users/liamr/Ap
Uniform:  5112.348879315299 | Probability Based 6373.100520775513
improvment: 1.246609077592738
PS C:\Users\liamr\OneDrive\Desktop\PJAS 2022> & C:/Users/liamr/Ap
Uniform:  5057.962028395162 | Probability Based 6035.573936831037
improvment: 1.1932817808729301
PS C:\Users\liamr\OneDrive\Desktop\PJAS 2022> & C:/Users/liamr/Ap
Uniform:  4946.515955278075 | Probability Based 6360.575936300846
improvment: 1.285869891820308
PS C:\Users\liamr\OneDrive\Desktop\PJAS 2022> & C:/Users/liamr/Ap
Uniform:  5018.130273607793 | Probability Based 6297.528196652417
improvment: 1.2549551034522661
PS C:\Users\liamr\OneDrive\Desktop\PJAS 2022>
```

The function, created before section one & two, is used as comparative metric to quantify the improvement of the specified distribution function to the uniform standard. The section three comparison indicated a 124.1% improvement of the new distribution method to its predecessor.

Summary

- I was able to successfully create a machine learning algorithm which took input from a telemarketing dataset, and to a reasonable degree was able to predict the percent-likelihood that an individual would subscribe to a bank term deposit.
    - In order to do this I had to encode the data (which we were able to visualize using T-SNE) and process it through a Logistic Regression algorithm.
- Having obtained the ability to predict these likelihoods, I implemented a real-world application of the first section by creating a distribution function which factored likelihoods to produce a budget allocation such that every individual would be advertised to as cost-effectively as possible.
- Finally, I used a separate function to compare the distribution methods. The output showed a great improvement (124%) from the traditional distribution method, to the adapted one.

Conclusion

- The Logistic Regression algorithm performed well showing very decent discrimination measure indicating that the variables/categories present do correlate to ultimate decision.
- After doing data visualization with T-SNE, it was clear that the data does not have a very good separation between the classes, as the two classes were highly overlapping even after trying different parameters.
- The mathematical function created during the second portion of this case-study supports the notion that even under specialized conditions (where each individual has an effect), the distribution of budget would still follow the general outline presented in a gaussian distribution.
- I believe that my initial hypothesis was accurate.

Real World Application

- Avertisment, and subsequent branches like telemarketing, play a crucial role in businesses and institutions. Advertising works to drive business growth, amplify small business marketing, and helps reach the right audience with positive, targeted messaging that converts potential customers into paying customers. The research I have conducted would benefit all manners of businesses and might even act as a stepping stone for more theoretical economic/marketing research.

Errors & Further Study

Errors:

- The AUC score presented only a moderate discrimination measure as a result of the complexity of predict human behavior and the data's separation between classes

Further study:

- With more time to conduct research, I would possible compare various response-coding algorithms in an effort to find the ideal approach and perfect the crucial first step
- Additionally, I would like to directly correlate the first and second part of my project such that the percentage-likelihoods from the banking data set will flow directly into the distribution function

Citations

- https://archive.ics.uci.edu/ml/datasets/bank+marketing
- https://www.investopedia.com/terms/a/advertising-budget.asp
- https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc
- http://hyperphysics.phy-astr.gsu.edu/hbase/Math/gaufcn.html
- https://www.forbes.com/sites/forbesbusinesscouncil/2022/01/18/how-machine-learning-is-shaping-the-future-of-advertising/?sh=3f84cc301361
- https://www.linkedin.com/pulse/importance-telemarketing-lead-generation-roy-heaton
- https://www.digitalthirdcoast.com/blog/marketing-math
- https://www.analyticsvidhya.com/blog/2021/09/guide-for-building-an-end-to-end-logistic-regression-model/

Thank You For Listening!