Coupon usage prediction on In-Vehicle Recommendation systems

Group 5

Anushka Hegde

Yanming Liu

Krishna Hemant

hegde.anu@northeastern.edu

liu.yanmi@northeastern.edu

durgasharan.k@northeastern.edu

Signature of Student 1: *Anushka_Hegde*

Signature of Student 2: *Yanming Liu*

Signature of Student 3: *Krishna_Hemant*

Submission Date: 8/08/2022

Abstract

Coupons are seen almost everywhere these days, from online clothing websites to travel agencies, food delivery apps and third party websites or applications hosting them. Earlier, basic models or mostly experience and pure domain knowledge were used in maximizing the coupon usage to boost revenue generation. As computation algorithms get more powerful everyday, companies are starting to rely on machine learning models which lend a data driven, unbiased lens to solve problems, such as prediction of coupon usage. This project focuses on a third party service, an in-vehicle system (like Uber) hosting the coupons on their service. Hosting coupons on another service can bring a lot of additional costs. Those expenses coupled with the initial marketing and planning costs of introducing these coupons can be a big financial burden on the company. The current Machine Learning algorithms can be a powerful tool in reducing these costs and targeting the right crowd to maximize coupon usage.

Introduction

Every user who rides the vehicle brings with them a series of data points which can be really helpful in predicting their usage of the displayed coupon. The purpose of this project lies in using all these available data points to build a robust machine learning model to classify if the user will accept the given coupon or not. The model will reduce the financial costs by a big margin by deciding if they should host the coupon or not to the given user based on their unique features. The coupons will be displayed on the application the user uses for the vehicle and they will be hosted on this application based on our prediction on their effectiveness. The company hosting the coupons will gain access to the user's information and our model will use this to gain actionable insights.

Data Source

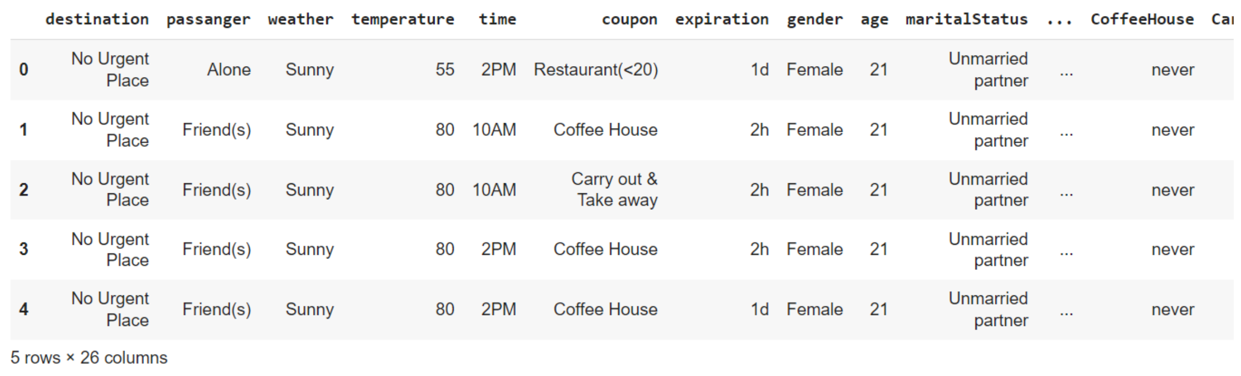
The data was taken from the UCI Machine learning repository
https://archive.ics.uci.edu/ml/datasets/in-vehicle+coupon+recommendation

Data Description and Cleaning

The dataset originates from the UCI Machine Learning Repository. By addressing surveys on Amazon Mechanical Turk, 12684 records with 26 attributes are acquired along with their missing values. All attributes are categorical variables, even age, which is usually encoded as a numeric. To be specific, the content for features can be split into four types:

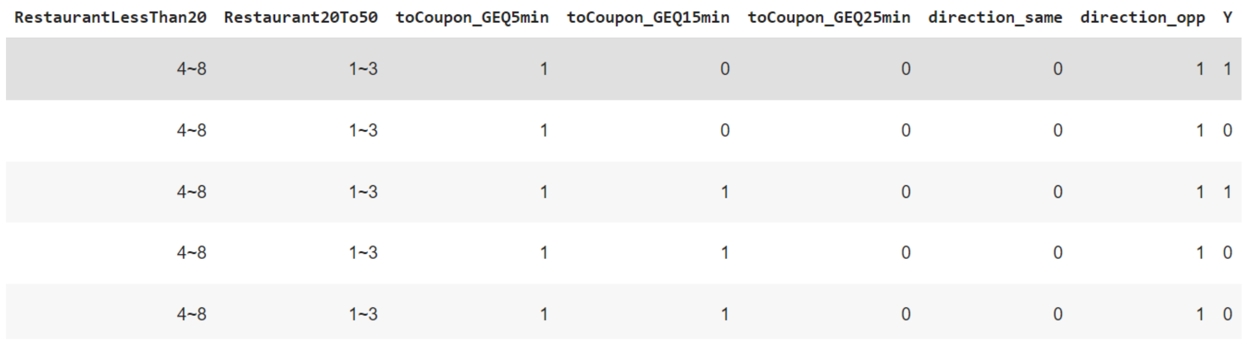
- **Travel info:(7 Features)** destination, passenger, weather, temperature, time, direction_same, direction_opp.
- **Demographic info:(8 Features)** like gender, age, marital status, child status, education, occupation, income, car possession type.
- **Coupon related info:(2 Features)** coupon type, expiration.
- **Purchase behavior info :(7 Features)** bar, coffeeHouse, carryAway, restaurant Less Than20, restaurant Less Than20, restaurant 20To 50, to Coupon GEQ15min, toCoupon GEQ25min.
- **Target:** Y: (1,0), 1 - Coupon acceptance, 2 - Coupon Rejection

Here are sample records from the in-vehicle coupon recommendation data set.



	destination	passanger	weather	temperature	time	coupon	expiration	gender	age	maritalStatus	...	CoffeeHouse	Ca
0	No Urgent Place	Alone	Sunny	55	2PM	Restaurant(<20)	1d	Female	21	Unmarried partner	...	never	
1	No Urgent Place	Friend(s)	Sunny	80	10AM	Coffee House	2h	Female	21	Unmarried partner	...	never	
2	No Urgent Place	Friend(s)	Sunny	80	10AM	Carry out & Take away	2h	Female	21	Unmarried partner	...	never	
3	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	2h	Female	21	Unmarried partner	...	never	
4	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	1d	Female	21	Unmarried partner	...	never	

5 rows × 26 columns



RestaurantLessThan20	Restaurant20To50	toCoupon_GEQ5min	toCoupon_GEQ15min	toCoupon_GEQ25min	direction_same	direction_opp	Y
4~8	1~3	1	0	0	0	1	1
4~8	1~3	1	0	0	0	1	0
4~8	1~3	1	1	0	0	1	1
4~8	1~3	1	1	0	0	1	0
4~8	1~3	1	1	0	0	1	0

Luckily, we got a clean data set from the UCI website. We checked for null and abnormal values and had to drop the car column as it had a lot of null values. Then we imputed missing values in certain columns with the most common occurrences. After this simple yet effective cleaning step, we move to model description to talk about the methods we have chosen and the reasons behind it.

Model Selection:

Four types of models will be fitted on train data and tested on validation data for the binary classification task (If the user will accept the coupon or not). First, we briefly introduce the models. Then, the reason and attention for each choice are illustrated.

Logistic Regression:

Logistic Regression aims to find the probability or propensity of a new record to belong to one class. This entails the purpose of binary classification. Similar to the linear regression method, it assumes that the predictors are not related and that there is no multicollinearity among predictor variables. We use mixed encoding(One hot + Ordinal) to represent the categorical variables and drop the first column for each dummy variable to avoid multicollinearity.

Advantages:

- Great for Binary classification with the sigmoid function
- Doesn't assume distribution for classes
- Trains well and is very fast on classifying unknown records

Disadvantages:

- Classifies linearly(Which is fine in our case)
- Requires variables to not have multicollinearity

Neural Networks:

Deep learning is a constantly evolving subfield of Machine learning. Deep learning analyzes the input in layer by layer manner, where each layer has different activation functions and progresses by extracting complex information about the input which normally cannot be done by the classical models. A Neural Network or an Artificial Neural Network is known for high predictive performance in classification and regression tasks by imitating biological properties of the brain.

We decided to go forward with the Keras framework from tensorflow to run a Neural Network on our dataset because of its high versatility and scalability.

Advantages:

- A highly complex and powerful model which can work on different data distributions for classification or regression tasks
- High performance and simple to implement
- Has a lot of features to tune the model and is highly scalable

Disadvantage:

- Model is a blackbox, hard to understand the workings of the model.
- Can take up a lot of CPU and GPU space

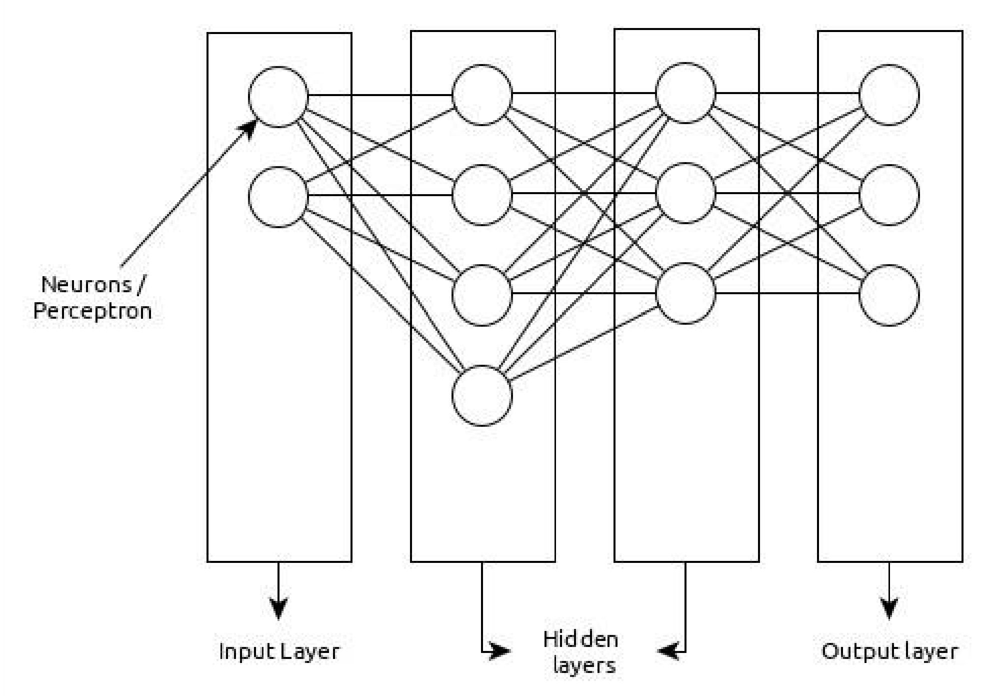


Fig 1: An example of a Neural Network Representation

Support Vector Machines(SVM):

SVM is a supervised machine learning algorithm which uses support vectors and builds a maximum margin hyperplane to classify the data points to the target variable. We have employed two of the SVM methods listed below to our dataset

1. Hard Margin SVM
2. Soft Margin SVM (Allows misclassification)

Advantages:
- Works well on high dimensional data
- Great for binary classification
- Kernel functions allow for complex datasets to be classified

Disadvantages:
- Soft margin SVM is very slow and takes up a lot of training time
- Classifies through geometry and is not probabilistic

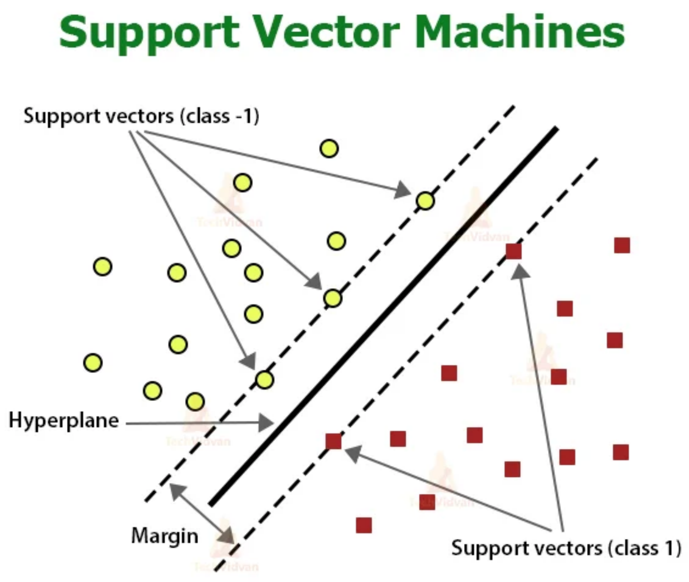


Fig 2: Example of a support vector classification

Naive Bayes:

Naive Bayes is simple yet one of the strongest probabilistic classifiers in Machine learning. It uses Bayesian probability to classify and it assumes independence of all features. and it was an easy choice as it works best on categorical data and our dataset was 90% categorical.

Advantages:
- Simple and would require no iterations, the probabilities would be directly computed
- Works great if independence of features hold
- Simple to implement and fast to train data
- A go-to choice as our dataset is mostly categorical

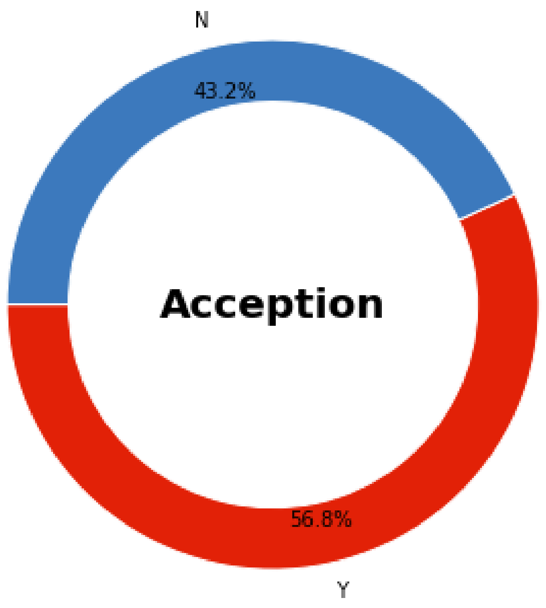
Disadvantages:

- Has problems with zero probabilities or zero frequencies
- Usually a dataset has some dependency, which the algorithm can be sensitive to

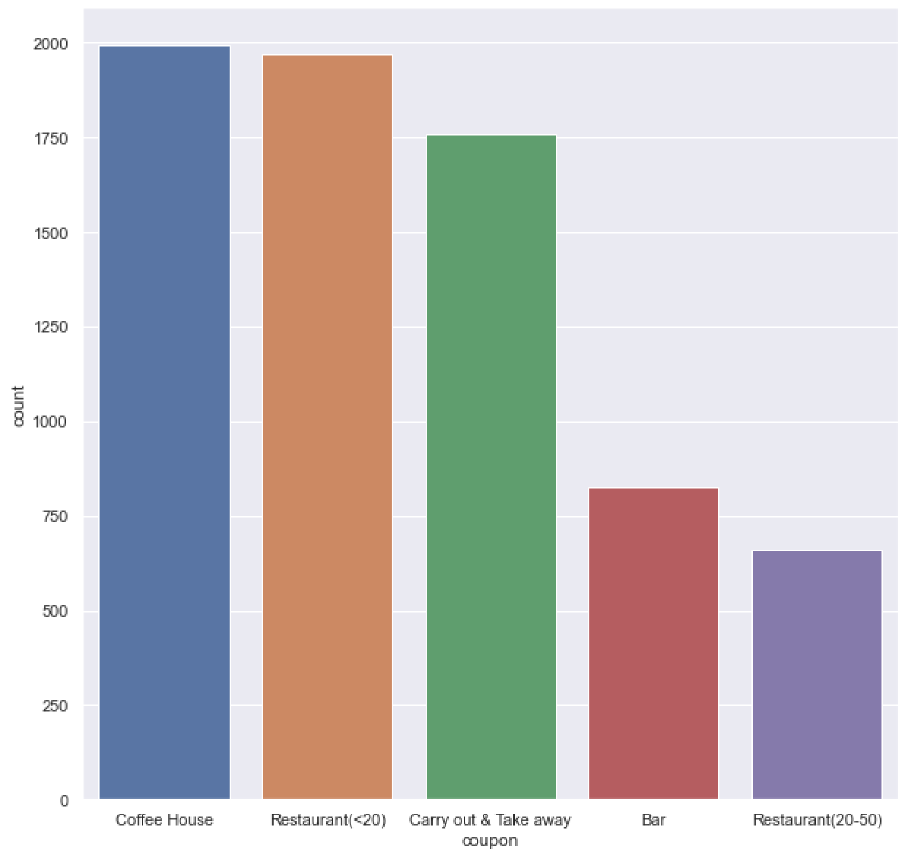
After discussing the machine learning algorithms we are considering, we move further to do a Exploratory data analysis of our dataset to understand it further and apply feature engineering to it.

Exploratory Data Analysis(EDA):

Our first step in Exploring our dataset would be to check the distribution of our target data with respect to our classes. We use python's matplotlib package to see a quick pie chart of our coupon acceptance and rejection percentages. Our dataset looks quite balanced and we don't have to worry about sampling problems.



And then we take a look at the coupons which are accepted the most,

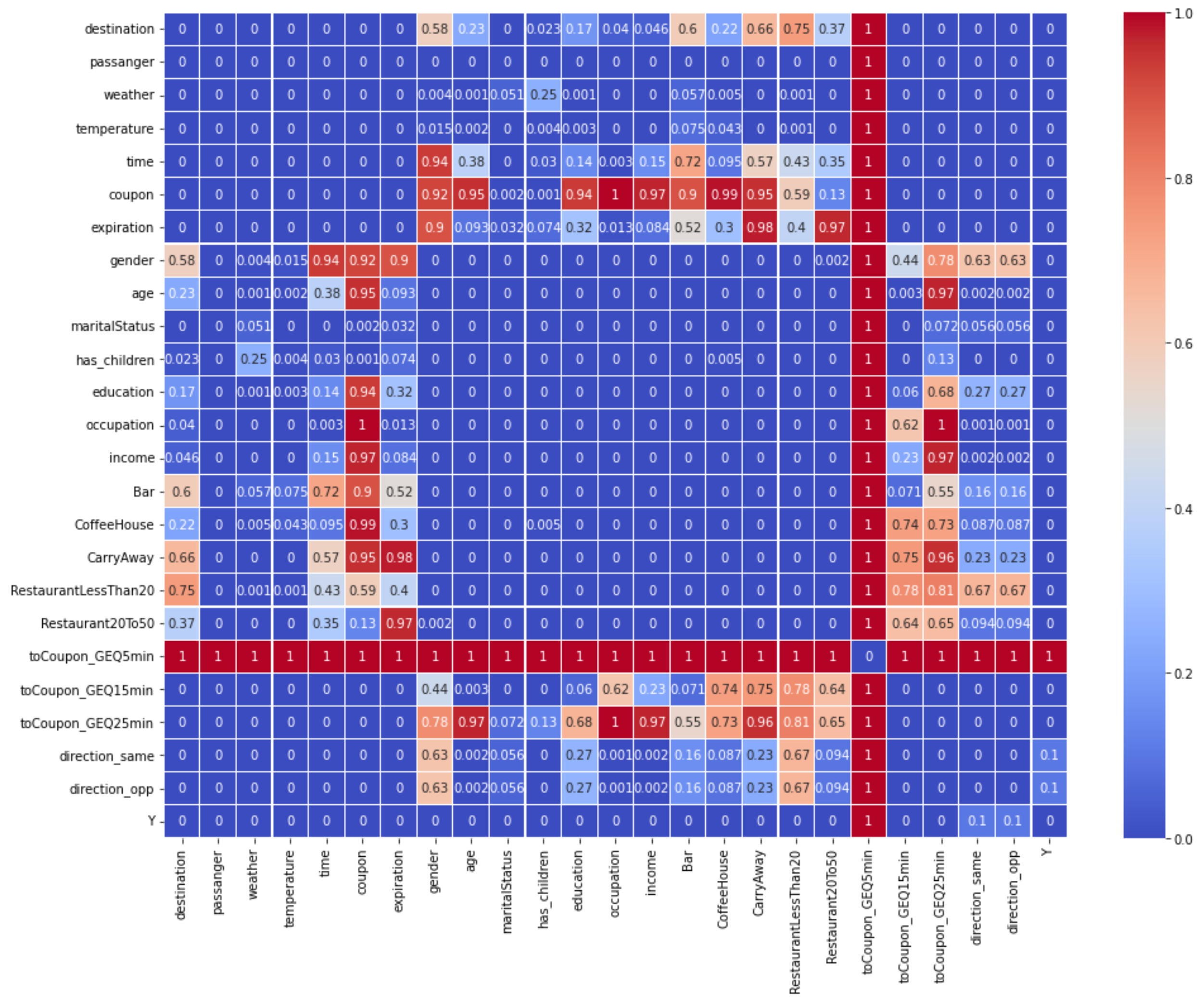


<u>Chi-Square Test to explore Correlation among variables:</u>

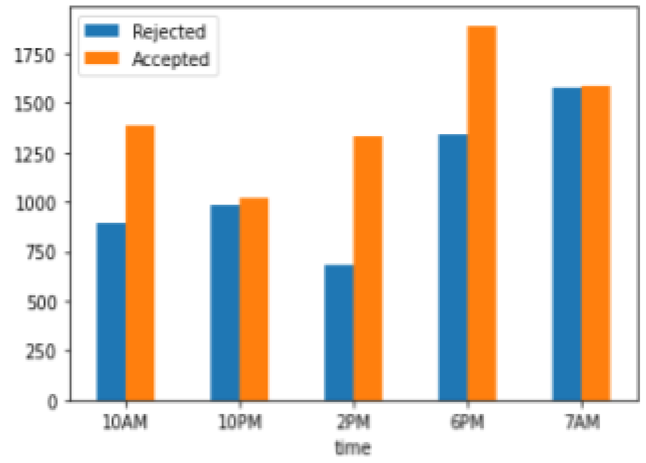
Next, we conduct a chi-square test to check the independence of the features and the target.

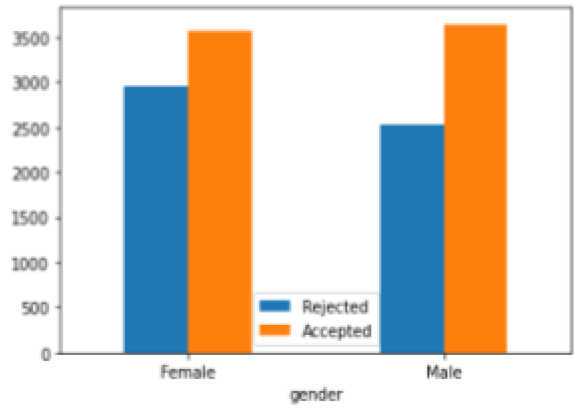
- If the p-value obtained is greater than 0.05, we fail to reject null hypothesis H0 and conclude that the predictor is not correlated to the target variable treatment.
- If the p-value obtained is less than 0.05, we reject null hypothesis H0 and conclude that the categorical predictor is correlated to the target variable treatment

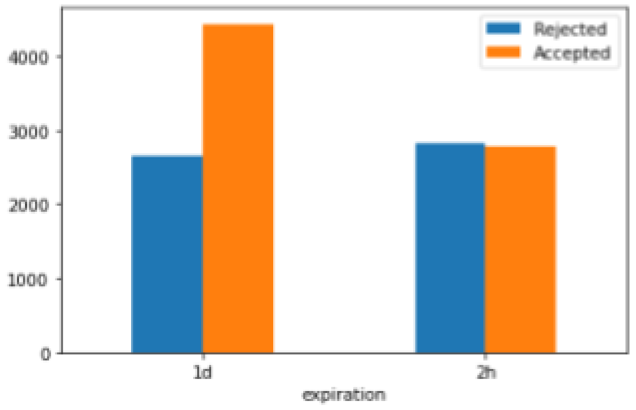
We conduct Chi-square test and plot p-values using heatmap to verify the independence between features and target. From the bottom line of the heatmap, we can tell the p values between toCoupon_GET5min, destination_same, destination_opp and Y target are 1, 0.1, 0.1. If our alpha limit is set as 0.05, that means if the p-value is below 0.05, we will consider the feature and target variable to be dependent whereas for a p-value above 0.05, the variables are independent of one another. This provides the reference for feature selection.

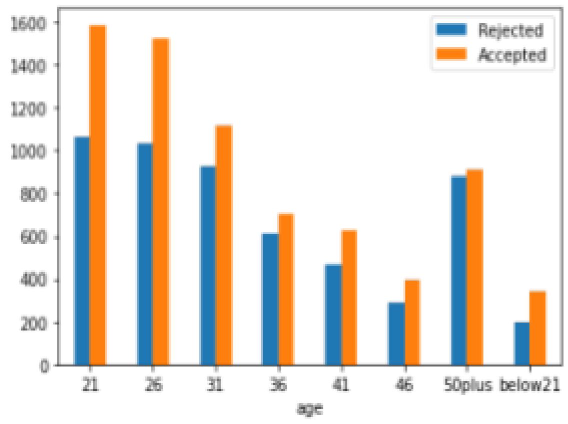


How Categories within features behave with respect to the target:

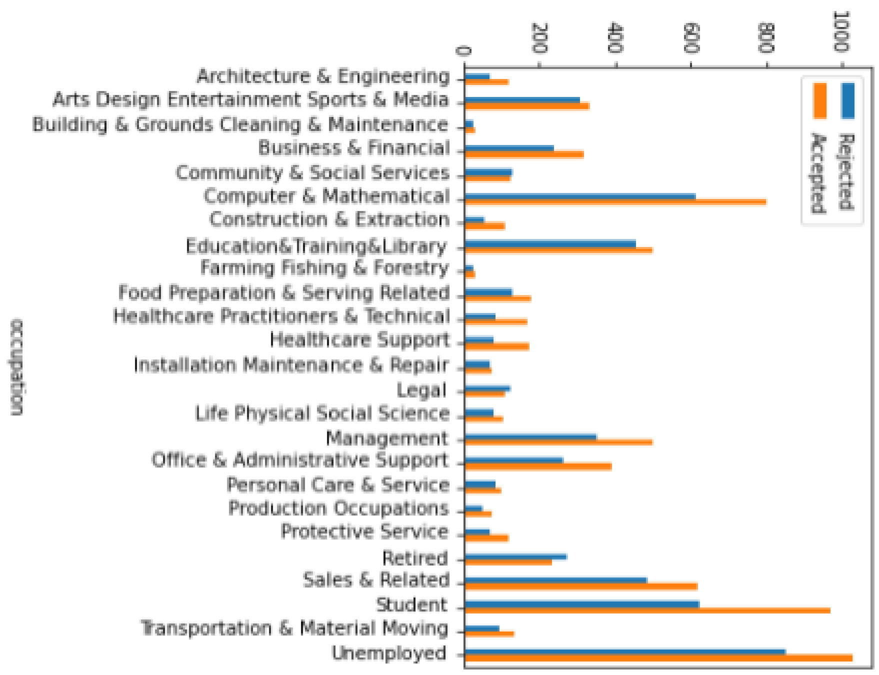




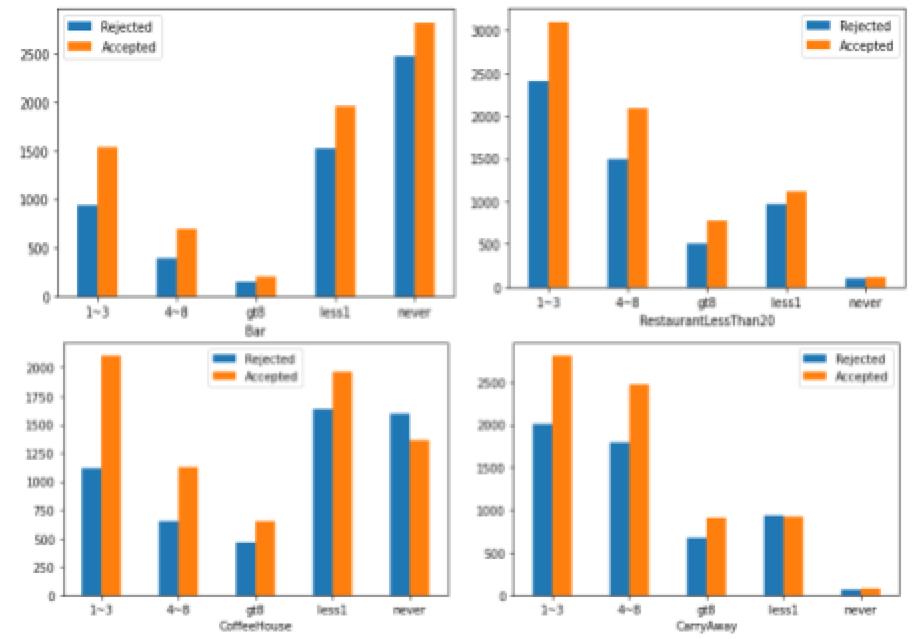




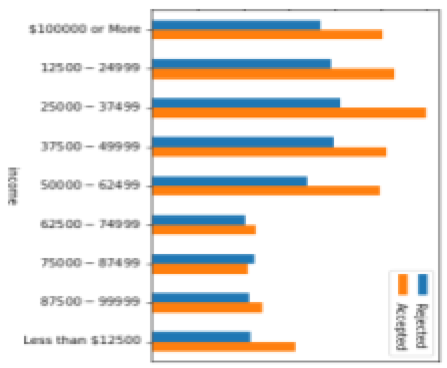
Looking at time, we notice that people usually accept coupons during the evening hours. With respect to gender, we don't see a significant statistic. There seems to be a much higher acceptance rate for coupons with a longer (24 hr) expiration date, maybe because people have more time to think about it. Younger people (age 21 to 31) seem to be a better target for these coupons, could be a generational thing.

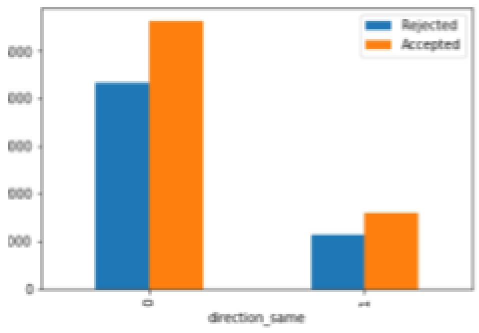


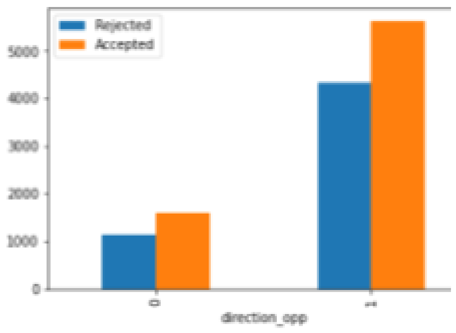
Here we notice unemployed people, Computer & Mathematical, and Students have the highest acceptance rates for these coupons, all of them trying to save money and could be good targets for the coupons. Interesting to see Management and sales and related people to have high acceptance rates too, maybe with the prior knowledge of marketing and business tactics.



Here we notice that the highest coupon acceptance is for people who never go to a bar, which correlates because the bar coupons have the least acceptance rate. People who visit a restaurant / coffee shop/ get a take out seem to give the highest acceptance rates for these coupons







From the income feature, we see that people within the income range of 12,500 to 62,499 have the highest usage and acceptance rates, with a low acceptance rate from people with income from the range 62,000 -99,999. An interesting observation here is noticing higher coupon acceptance rates from people with income above 100,000, which is unusual. Direction_same and direction_opp are the same column technically, so we drop one of them.

Feature selection

Based on the chi-square test done above, we look at the p-values from the heatmap and decide to drop 3 columns as they do not meet the confidence conditions. The columns dropped are,

- toCoupon_GEQ5min
- direction_same
- direction_opp

Feature Engineering

For our categorical variables, we attempted one hot encoding, ordinal encoding without mapping of orders, ordinal encoding with mapping of orders, target encoding and CatBoost encoder. Before testing the models, the dataset was cleaned and transformed by splitting the dataframe into train data, validation data, and test data with a ratio 6/2/2 as per the recommendations of Andrew Ng's online machine learning course.

The implementation of different encoding styles are explained further as follows,

Ordinal Encoding:

Ordinal encoding converts each label into integer values and the encoded data represents the sequence of labels. Here, we have tried two ways, package and manual implementation. The package is *category_encoder* to assign values for categories in features randomly, manual implementation maps the relationships between categories and integer values with prior knowledge. Furthermore, we can ordinal encode to all features or some features, like age, education, incoimme, Bar, CoffeeHouse, CarryAway, RestaurantLessThan20, Restaurant20To50.

One hot encoding:

A one-hot is a group of bits among which the legal combinations of values are only those with a single high (1) bit and all the others low (0) .It is usually used for transforming categorical variables into values, which can be suitable for machine learning models. Several ways can implement this encoding style, like get_dummies, OneHotEncoder from category_encoder and sklearn.preprocessing package.

Mixed encoding:

Mixture encoding is our custom name for the combination of one hot encoding and ordinal encoding. We implement ordinal encoding as both ordered and random assignment of numbers(which is done by ordinal_encoder from the category encoder package), and combine it with encoding from the one hot encoder function from the same package.

Target encoding:

In target encoding, features are replaced with a calculation that is a blend of posterior probability of the target given particular categorical value and the prior probability of the target over all the training data. The benefits of target encoding is that it doesn't add to the dimensionality of the data. All the values would be transformed in the range of 0,1 which helps us avoid extra steps for data normalization.

Embedded(or Learned embedded) encoding:

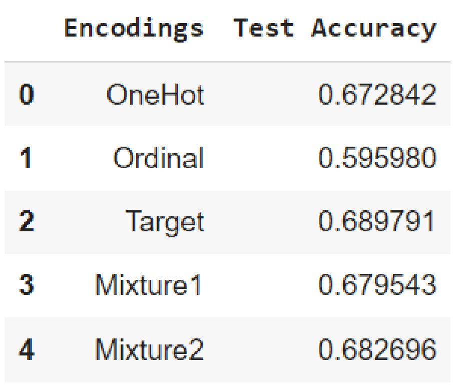
This encoding is a distributed representation of categorical data. Each category is mapped to a vector and the properties of the vector are learned through training a Neural Network (Keras in our case). The vectors are a representation of the categories and the closely related ones form a cluster. This technique allows for the benefits of ordinal and one hot encoding. It unfortunately cannot be used for all the models as the dimensions of data change and the dimensions of y_label won't match the ones with the X_label. For instance, cannot be used with logistic regression.

Our choice of encoding(and reasons)

The encoding styles we tried above,

- Ordinal Encoding
- One hot Encoding
- Mixed Encoding
- Target Encoding
- Embedded Encoding

To summarize, we first selected the features that are related to coupon acceptance based on the Chi-Square test. Then, we tested the classification accuracy based on logistic regression(selected as baseline model) for all the different encoded data. As for one hot encoding and mixture encoding, they both work well. Though we got the highest accuracy on target encoding, it performs badly in neural networks, probably because of overfitting the distribution and neural networks also being really complex in its architecture .The prior knowledge on the columns help us originally encode it with custom assignment and it helps us give an increased accuracy. We also explore the embedded encoding and apply it to neural networks, however, we can not apply it into other classical algorithms such as the logistic regression, naive bayes or SVM as the mismatch of shrinked records and y labels. Considering robustness, applicability, accuracy, we will choose the mixture encoding with custom values for categories(using domain knowledge) as our general encoding style for testing the performance of other models.



	Encodings	Test Accuracy
0	OneHot	0.672842
1	Ordinal	0.595980
2	Target	0.689791
3	Mixture1	0.679543
4	Mixture2	0.682696

Full implementation of models with evaluation metrics:

In this section, we will implement all the models with python code written with the help of libraries numpy and pandas without using the packages from scikit learn for model implementation. Then we will evaluate the models based on accuracy metrics and will measure time and space complexity used by the models.

Logistic Regression

We manually implement the logistic regression with stochastic gradient descent in mixture encoding, then obtain the train, validation, test accuracy as a baseline using some fixed parameters. Then, improve them on classification accuracy with different learning rates, tolerance, maxIteration, batch size. In this segment, we chose to fix all parameters barring one for each parameter instance. Thus, the optimal values of all parameters were iteratively achieved. Next, we validate the effectiveness of our manual code with tuned parameters compared to the accuracy of the LogisticRegression package.

With fixed parameters, learningRate = 0.00001, tolerance = 0.0001, maxIteration = 100, tolerance and maxIteration are the same as the default values that sklearn.linear_model.LogisticRegression package provides. The potential value choice for parameters are as follows,

```
learning_rate = [0.001, 0.005, 0.01, 0.1, 0.5]
Tolerance = [0.001, 0.005, 0.01, 0.1, 0.5]
max_Iterations = [100, 200, 500, 1000, 5000]
batch_sizes = [32, 64, 128, 512, 1028]
```

We got the training, validation, and test accuracy under the tuned parameters as the below snip shows in order.

Mixture: Accuracy for best parameters, learning rate 0.001000, tolerance 0.001000, max iteration 1000, batchSize 128

```
0.6758212877792379
0.6823019314150571
0.6720536066219945
```

The following is the validation and test accuracy using sklearn.logisticRegression package, which is quite close to the one that we implement manually. That means our code for manually implementing the logistic regression is correct.

```
0.6826960977532519
0.6767836026803311
```

Deep Neural Network

This section tunes the parameters on validation accuracy for deep neural networks. It aims to solve the following questions, what is the baseline of validation accuracy of deep neural networks? What are the hyper-parameters? How to tune them? What is the test accuracy with tuned parameters?

For the baseline of deep neural networks. We just have one softmax layer to map the data with the classification result with fixed parameter values. Here are parameters we consider in the following improvement: number of layers, number of hidden neurons, dropout percent, optimizers, number of epochs, batch size, learning rate and regularization type. Their original value for the validation accuracy can be found as follows. The value can be chosen arbitrarily here, just serving the function to know what the deep neural network performs.

```
#  parameters
n_hiddens = 128
dropout = 0
default_optimizer = 'SGD'
num_epochs = 200
batch_size = 128
learning_rate = 0.001
regtype = ''
verbose = 0
n_classes = 2
validation_split = 0.2
```

In later improvement, for each time, we will tune one parameter for the best validation accuracy, and use that parameter for another parameter tuning. Like accuray increments, here are some choices for the parameters.

```
Dropout = [0.1, 0.2, 0.3, 0.4, 0.5]
optimizers = ['RMSProp', 'Adam']
num_epochs = [25, 50, 100, 200, 300]
batch_size = [32, 64, 128, 256, 512]
learning_rate = [0.001, 0.005, 0.01, 0.1, 0.5]
n_hiddens = [128, 256, 512, 1024, 2048]
regtype = ['l1', 'l2']
```

Finally, we will get a set of tuned parameters.

```
Test accuracy: 0.735120 for Neural network with hidden layers, when dropout 0.300000, optimizers: Adam, epoch:50, batch_size:32, learning rate 0.001000, n_hidden:2048.
```

Support Vector Machine (SVM)

Support Vector Machine works as a powerful learning algorithm that can capture the non-linearity relationship. What is hard margin SVM and soft-margin SVM? What are the hyper-parameters here? How to tune them?

In our report, we try hard-margin SVM and soft-margin svm that is optimized by smo and optimize.minimize(). In hard-margin SVM, we turn the hyper-parameters such as epochs, learning rate and lambda. Compared to the hard-margin SVM, soft-margin one can tolerate more classification errors to capture non-linearity relationships underneath the data. We tried two ways, smo and minimize.minimize package to get the optimized coefficients. Both ways run slow and we just commit them for a normal run through for the whole notebook. In the latter industrial setting, we can play with them using GPU.

For mixture encoding - 2, the baseline of test accuracy for hard-margin accuracy is,

```
0.6720536066219945
```

For one-hot encoding, the baseline of test accuracy for hard-margin accuracy is,

```
0.6767836026803311
```

We tune the following parameters and here are their values using validation accuracy increment strategy to pick the best parameter.

```
epochs = [25, 50, 100, 200, 300, 1000]
learning_rate = [0.00001, 0.0001, 0.001, 0.005, 0.01]
lamda = [0.0001, 0.001, 0.01, 0.1, 1]
```

Here are the best parameters in this trial with a test accuracy of 0.6736302719747733.

Naive Bayes

Naive bayes are generally used for classification problems. We try to answer what is the difference between naive bayes and gaussian naive bayes? How to implement the gaussian naive bayes from scratch?

In this section, we implemented the gaussian naive bayes and got test accuracy on our data, which further deepened our understanding of gaussian naive bayes especially in high-dimensional features.

Here is the training, validation and test accuracy for gaussian naive bayes as follows,

```
0.6415243101182655
0.637761135199054
0.6357903035080804
```

Model Performance Comparison and Results:

We will measure the model performance by three metrics, like test accuracy, time consumption and space usage on previous models.

Accuracy

	Models	Train Accuracy	Validation Accuracy	Test Accuracy
0	Logistic Regression - Baseline	0.556242	0.566417	0.566023
1	Logistic Regression - Tuned	0.675821	0.682302	0.672054
2	Neural Network - Baseline	0.680815	0.675995	0.673630
3	Neural Network - Tuned	0.939290	0.735909	0.735120
4	Hard SVM - Baseline	0.686071	0.686244	0.672054
5	Hard SVM - Tuned	0.684888	0.687032	0.673630
6	Gaussian Naive Bayes	0.641524	0.637761	0.635790

Based on the accuracy metrics, we observe that the Neural Networks model outperforms all the other classifiers on the test and validation set. The performance of the tuned Hard Margin SVM is comparable with the untuned Neural Network.

Time Consumption

	Model	Processing Time(s)
0	Logistic Regression	49.681866
1	Neural Network	883.636559
2	Hard SVM	135.805133
3	Gaussian Naive Bayes	339.217640

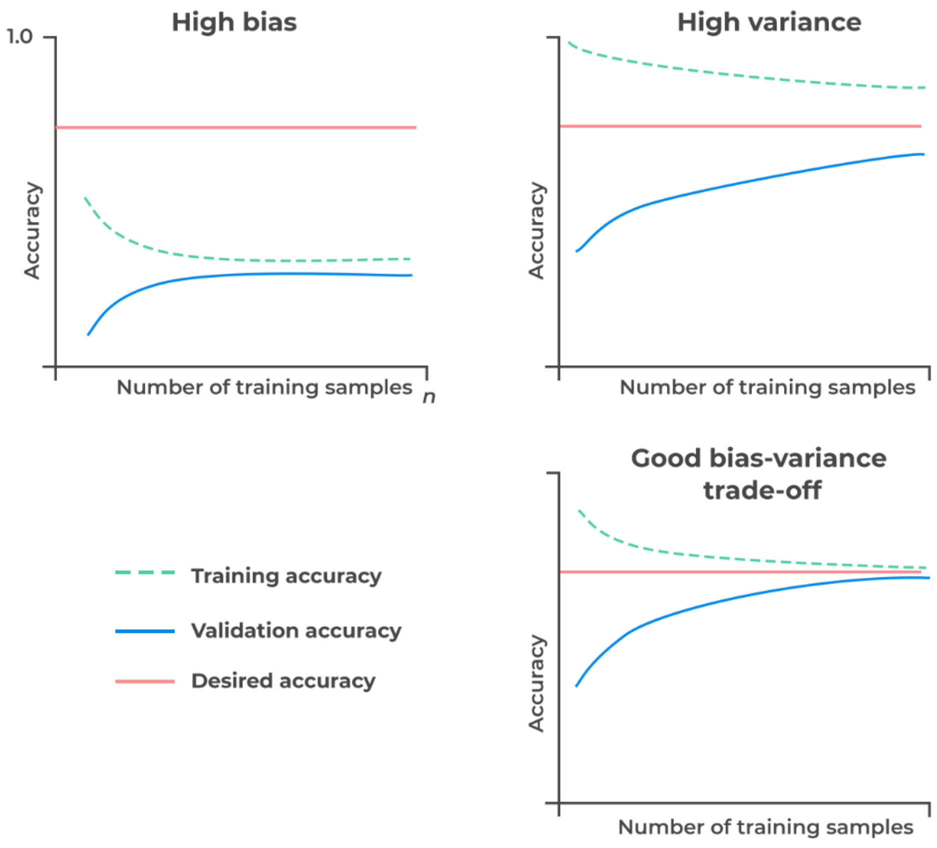
Space Usage

```
                  Model  Peak Memory(B)
0   Logistic Regression        13922189
1        Neural Network        13993211
2              Hard SVM         4451905
3  Gaussian Naive Bayes        13520765
```

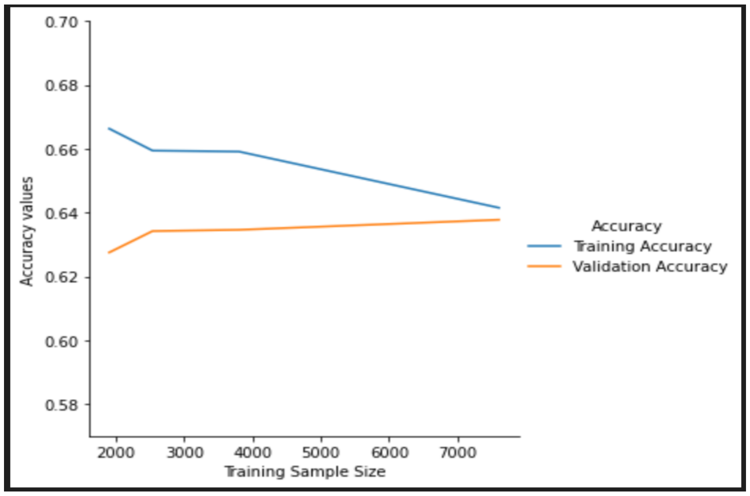
On the other hand, if we look at the run-time comparisons, Neural Networks take up the largest processing time. From the snapshot of the memory allocations for fitting and prediction, we can gauge the RAM usage of all models. Hard Margin SVM appears to be the most efficient of all the models.

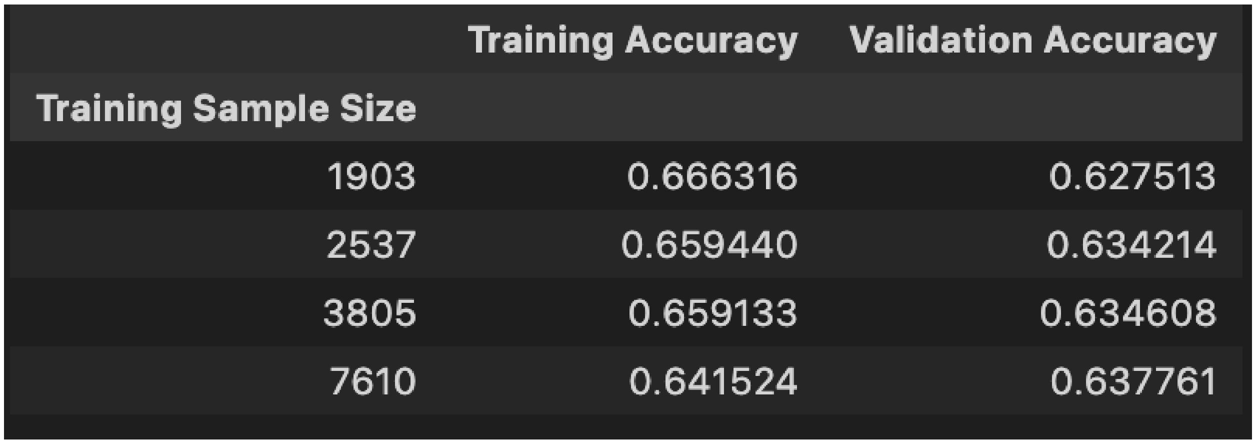
Bias - Variance Tradeoff (Using Learning Curve):

A learning curve shows the validation and training accuracy of a model for varying numbers of training sample sizes. It represents a tool to estimate how much the model benefits from adding more training data and whether the model suffers more from a bias error or a variance error. The figure below displays three possible scenarios. A model with good bias-variance trade-off will show a very small difference in the training and validation accuracy on convergence. While this small difference in convergence values is maintained in the model with a high bias learning curve, the overall performance (accuracy) is much lower than expected. Contrastingly, in the case of a high variance learning curve, since the model overfits to the data, the difference between the training and validation accuracy does not converge to a low value.



We created a learning curve using the Gaussian Naive Bayes model with multiple subsets of our training data with increasing sample size. The data sizes considered have been presented in the table below. We observe that the difference between the training and validation accuracies is reduced with each subset with the final convergence difference being 0.0038. However, since the overall performance of the Gaussian Naive Bayes model is quite low, we infer that this represents a high bias learning curve.





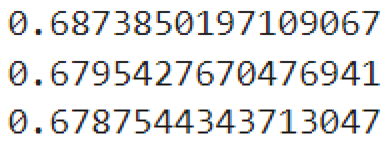
	Training Accuracy	Validation Accuracy
Training Sample Size		
1903	0.666316	0.627513
2537	0.659440	0.634214
3805	0.659133	0.634608
7610	0.641524	0.637761

To address high bias learning curves, it is recommended to increase the number of parameters or create new features. To verify this suggested improvement in accuracy, we experimented with the One Hot Encoded data set in Logistic Regression explored earlier, since the number of features exceeds the dimensions of the mixture encoded data. Train, Validation and Test accuracy was obtained in the following order.

For mixture encoding,

```
0.6758212877792379
0.6823019314150571
0.6720536066219945
```

For one-hot encoding,



```
0.6873850197109067
0.6795427670476941
0.6787544343713047
```

Due to lack of significant variation in performance, we decided to implement all the models with mixture encoded data as our main dataset of choice.

Discussion and Conclusion:

Based on the model selection criteria, we can choose the optimal models according to our preference for accuracy or efficiency. Since the overall performance of the tuned Neural Networks model is approximately 6% higher than the next best, when prioritizing high accuracy, this model will work well. When computational efficiency may be an issue, particularly in the case of large datasets, it may be wiser to go with the Hard Margin SVM classifier.

With this project, our goal was to build multiple classifiers capable of understanding customers' response to recommendations made in an in-vehicle setting, that promote coupons for local businesses. The coupons were targeted to the user in their specific context. Based on the data collected from the Amazon Mechanical Turk survey of 12684 data cases and the models that we trained and validated the data on, we opted for the Neural Network model to analyze if a customer is going to accept a coupon for a particular venue, considering various demographic and contextual attributes. This model could help companies looking to gain an overview of their key customer base and potentially assist in the assessment of cost-benefit analysis of future coupon marketing schemes. These models may also find application in pre-training networks for similar in-vehicle digital interventions [Ref] and other consumer behavior modeling programs.

GitHub : [link]