DATA SCIENCE PROJECT

Duong Thuy Le

Part A:

Part A.1: Describe the numbers below in a table:

Cross-validation Fold	Decision Tree	Logistic Regression
Fold 1	0.657	0.644
Fold 2	0.64	0.641
Fold 3	0.644	0.635
Fold 4	0.672	0.637
Fold 5	0.676	0.653
Fold 6	0.63	0.606
Fold 7	0.662	0.622
Fold 8	0.648	0.613
Fold 9	0.646	0.615
Fold 10	0.658	0.618
Average Error %	65.33	62.84
Std. Dev. Error %	1.37	1.48

Part A.4: If Telco company proceeded with a marketing campaign, where:

- The marketing material for each customer will cost $5 (including designing, printing, and mailing to each household). The cost of goods sold (COGS) to make each box is $200.
- The revenue from each product package charged per customer will be $500.

We can conclude the cost-benefit matrix as below:

	p	n
Y	b (Y, p) = \$295	c (Y, n) = -\$5
N	c (N, p) = \$0	b (N, n) = \$0

Based on the relation between the confusion matrix and the cost-benefit matrix, we will divide it into two main categories:

1) **Correct classifications:**

- Confusion matrix: true positives/cost-benefit matrix: *b* (**Y**, **p**): A *true positive* is a consumer who is correctly identified as being an actual buyer, which may allow for the delivery of customized recommendations, targeted offers, or other advantages that could enhance their interaction with the company. Since its benefits are to increase customer loyalty, repeat business, and favorable word-of-mouth, whereas extra cost (apart from marketing worth and COGS) for accurately recognizing a positive consumer is \$0. Thus, b (Y, p) = gross profit = \$500 (revenue) - \$200 (COGS) - \$5 (marketing cost) - \$0 (loss) = \$295.

- Confusion matrix: true negatives/cost-benefit matrix: *b* (**N**, **p**): A *true negative* is a consumer correctly identified as being actual churn, which helps the company cut down costs there and invest business on more potential customers. In addition, there is no cost spent to promote, so that we could classify this situation as a benefit to the company – *b* (**N**, **n**) = 0.

2) **Incorrect classifications:**

- Confusion matrix: false positives/cost-benefit matrix: *c* (**Y**, **n**): A *false positive* is a consumer predicted as a potential buyer, but that person does not respond. Since the company targets marketing on this customer, loss of revenue, a decline in customer loyalty, and negative word-of-mouth are possible outcomes if that customer is unsatisfied with the marketing activity. The benefit, in this case, is harmful. This is the company's cost, so *c* (**Y**, **n**) = - \$5 (marketing cost).

- Confusion matrix: false negatives/cost-benefit matrix: *c* (**N**, **p**): A *false negative* is a consumer predicted to be a churn, but in fact, that person could be a potential customer if the package has been offered. In this case, even if no money was spent, the company has lost an opportunity to acquire the product, so *c* (**N**, **p**) = 0.

Part A.5:

	RULES for identifying	Description in words
CHURN Segment 1	Entropy = 0.308, samples = 3.8%, value = [0.055,	The given leaf node corresponds to a subset of the churn dataset containing 3.8% of the total samples. The entropy value for this subset is 0.308, which indicates a moderate level of impurity or randomness in the subgroup. The "value" parameter represents the distribution of the target variable (i.e., churn or non-churn) within the subset. In this case, the "value" parameter indicates that out of the total number of instances in this subset, 5.5% (or approximately 0.055 times the number of instances) belong to the "churn" class, and 94.5% (or about 0.945 times the number of instances) belong to the "stay" class. Finally, the "class" parameter indicates this leaf

	0.945], class = stay	node's predicted or majority class label. In this case, the "stay" class is the majority class, as it has a higher percentage of instances in the subset than the "churn" class. In summary, the given leaf node represents a subset of the churn dataset with moderate impurity. Most instances belong to the "stay" class, and only a small proportion belongs to the "churn" class. This information can help understand the data's characteristics and interpret the results of a machine learning model trained on this dataset.
CHURN Segment 2	Entropy = 0.654, samples = 3.9%, value = [0.168, 0.832], class = STAY	The given leaf node corresponds to a subset of the churn dataset containing 3.9% of the samples. The entropy value for this subset is 0.654, which indicates a relatively high level of impurity or randomness in the subgroup. In this case, the "value" parameter indicates that out of the total number of instances in this subset, 16.8% (or approximately 0.168 times the number of instances) belong to the "churn" class, and 83.2% (or about 0.832 times the number of instances) belong to the "stay" class. For the "class" parameter, the "stay" class is still the majority class, as it has a higher percentage of instances in the subset than the "churn" class. In summary, the given leaf node represents a subset of the churn dataset with relatively high impurity, where most instances still belong to the "stay" class. Still, there is a higher proportion of instances belonging to the "churn" class than in the previous example.
CHURN Segment 3	Entropy = 0.997, samples = 7.2%, value = [0.468, 0.532], class = STAY	The given leaf node corresponds to a subset of the churn dataset containing 7.2% of the samples. The entropy value for this subset is 0.997, which indicates a very high level of impurity or randomness in the subset. In this case, the "value" parameter indicates that out of the total number of instances in this subset, 46.8% (or approximately 0.468 times the number of instances) belong to the "churn" class and 53.2% (or approximately 0.532 times the number of instances) belong to the "stay" class. For the "class" parameter, the "stay" class is still the majority class, as it has a higher percentage of instances in the subset compared to the "churn" class. In summary, the given leaf node represents a subset of the churn dataset with very high impurity, with a nearly equal distribution of instances belonging to the "stay" and "churn" classes.
CHURN Segment 4	Entropy = 0.96, samples = 14.9%, value = [0.168, 0.382], class = LEAVE	The given leaf node corresponds to a subset of the churn dataset containing 14.9% of the total number of samples. The entropy value for this subset is 0.96, which indicates a very high level of impurity or randomness in the subset. In this case, the "value" parameter indicates that out of the total number of instances in this subset, 16.8% (or approximately 0.168 times the number of instances) belong to the "churn" class and 38.2% (or approximately 0.382 times the number of instances) belong to the "leave" class. For the "class" parameter, the "leave" class is the majority class, as it has a higher percentage of instances in the subset compared to the "churn" class.

		In summary, the given leaf node represents a subset of the churn dataset with very high impurity, where most instances belong to the "leave" class, and only a small proportion belongs to the "churn" class.
CHURN Segment 5	Entropy = 0.73, samples = 22.2%, value = [0.796, 0.204], class = LEAVE	The given leaf node corresponds to a subset of the churn dataset containing 22.2% of the total number of samples. The value of entropy for this subset is 0.73, which indicates a relatively high level of impurity or randomness in the subset. In this case, the "value" parameter indicates that out of the total number of instances in this subset, 79.6% (or approximately 0.796 times the number of instances) belong to the "leave" class and 20.4% (or approximately 0.204 times the number of instances) belong to the "stay" class. For the "class" parameter, the "leave" class is the majority class, as it has a much higher percentage of instances in the subset compared to the "stay" class. In summary, the given leaf node represents a subset of the churn dataset with relatively high impurity, where most instances belong to the "leave" class, and only a small proportion belongs to the "stay" class.
CHURN Segment 6	Entropy = 0.685, samples = 21.5%, value = [0.182, 0.818], class = STAY	The given leaf node corresponds to a subset of the churn dataset containing 21.5% of the total number of samples. The value of entropy for this subset is 0.685, which indicates a relatively high level of impurity or randomness in the subset. In this case, the "value" parameter indicates that out of the total number of instances in this subset, 18.2% (or approximately 0.182 times the number of instances) belong to the "churn" class and 81.8% (or approximately 0.818 times the number of instances) belong to the "stay" class. For the "class" parameter, the "stay" class is the majority class, as it has a much higher percentage of instances in the subset compared to the "churn" class. In summary, the given leaf node represents a subset of the churn dataset with relatively high impurity, where most instances belong to the "stay" class, and only a small proportion belongs to the "churn" class.
CHURN Segment 7	Entropy = 0.902, samples = 1.3%, value = [0.318, 0.682], class = STAY	The given leaf node corresponds to a subset of the churn dataset containing 1.3% of the total number of samples. The value of entropy for this subset is 0.902, which indicates a relatively high level of impurity or randomness in the subset. In this case, the "value" parameter indicates that out of the total number of instances in this subset, 31.8% (or approximately 0.318 times the number of instances) belong to the "churn" class and 68.2% (or approximately 0.682 times the number of instances) belong to the "stay" class. For the "class" parameter, the "stay" class is the majority class, as it has a much higher percentage of instances in the subset compared to the "churn" class. In summary, the given leaf node represents a subset of the churn dataset with relatively high impurity, where most instances belong to the "stay" class, and only a small proportion belongs to the "churn" class

CHURN Segment 8	Entropy = 0.976, samples = 4.7%, value = [0.409, 0.591], class = STAY	In this case, there are 4.7% of the total number of data points in the dataset that belong to this leaf node. The entropy value of 0.976 indicates that there is a high degree of uncertainty or randomness in the data at this leaf node. The model predicts that there is a 40.9% probability that the customer will not churn (class STAY) and a 59.1% probability that the customer will churn. For the "class" parameter, the predicted class is STAY, since it has the highest predicted probability. Overall, this leaf node suggests that there is a significant degree of uncertainty or randomness in the data at this point, and the model predicts a higher probability of churn for the customers in this node. However, there is still a notable chance that these customers will stay, as indicated by the 40.9% probability of the STAY class.
CHURN Segment 9	Entropy = 0.975, samples = 2.5%, value = [0.594, 0.406], class = LEAVE	In this case, there are 2.5% of the total number of data points in the dataset that belong to this leaf node. The entropy value of 0.975 indicates that there is a high degree of uncertainty or randomness in the data at this leaf node. The model predicts that there is a 59.4% probability that the customer will churn (class LEAVE) and a 40.6% probability that the customer will not churn. For the "class" parameter, the predicted class is LEAVE, since it has the highest predicted probability. Overall, this leaf node suggests that there is a high degree of uncertainty or randomness in the data at this point, and the model predicts a higher probability of churn for the customers in this node. The predicted class is LEAVE, which means that the model predicts that these customers are likely to churn. However, there is still a notable chance that these customers will stay, as indicated by the 40.6% probability of the STAY class.
CHURN Segment 10	Entropy = 0.693, samples = 3.4%, value = [0.814, 0.186], class = LEAVE	In this case, there are 3.4% of the total number of data points in the dataset that belong to this leaf node. The entropy value of 0.693 indicates that there is a moderate degree of uncertainty or randomness in the data at this leaf node. The model predicts that there is an 81.4% probability that the customer will churn (class LEAVE) and an 18.6% probability that the customer will not churn. For the "class" parameter, the predicted class is LEAVE, since it has the highest predicted probability. Overall, this leaf node suggests that there is a moderate degree of uncertainty or randomness in the data at this point, and the model predicts a higher probability of churn for the customers in this node. The predicted class is LEAVE, which means that the model predicts that these customers are likely to churn. However, there is still a chance that these customers will stay, as indicated by the 18.6% probability of the STAY class.

As a business manager, the best segment to focus on reducing churn would be the **CHURN segment 5** with an entropy of 0.73, a sample size of 22.2%, and a value of [0.796, 0.204] with a class of LEAVE. This segment has high entropy, meaning there is a lot of uncertainty or randomness in the data. It also has a significant sample size, meaning it is a substantial portion of the customer base. However, the value of [0.796, 0.204] suggests that a large part of the customers in this segment are likely to leave, making it a high-risk feature. As a business manager, it would be essential to focus resources on reducing churn in this segment by identifying why customers are leaving and developing strategies to address those reasons.

By focusing on reducing churn in this high-risk segment, the business can have a significant impact on reducing overall churn and retaining valuable customers.

Appendix A.2: Decision Tree cross-validation notebook:

```
import pandas as pd
import numpy as np
from numpy import mean
from numpy import std
import os
from sklearn.model_selection import KFold, StratifiedKFold, cross_val_score
from sklearn import linear_model, tree, ensemble
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
import math
import matplotlib.pylab as plt
import seaborn as sns
```

```
path = "./churn2.csv"
df = pd.read_csv(path)[["COLLEGE", "INCOME", "OVERAGE", "LEFTOVER", "HOUSE","HANDSET_PRICE","OVER_15MINS_CALLS_PER_MONTH","AVERAGE_CALL_DURATION","REPORTED_SATISFACTION","REPORTED_USAGE_LEVEL","CONSIDERING_CHANGE_OF_PLAN","LEAVE"]].dropna()
# Take a look at the data
df.head(5)
```

Out[78]:

	COLLEGE	INCOME	OVERAGE	LEFTOVER	HOUSE	HANDSET_PRICE	OVER_15MINS_CALLS_PER_MONTH	AVERAGE_CALL_DURATION	REPORTED_SATISFACTION	REPORTED_USAGE_LEVEL	CONSIDERING_CHANGE_OF_PLAN	LEAVE
0	zero	31953	0	6	313378	161	0	4	unsat	little	no	STAY
1	one	36147	0	13	800586	244	0	6	unsat	little	considering	STAY
2	one	27273	230	0	305049	201	16	15	unsat	very_little	perhaps	STAY
3	zero	120070	38	33	788235	780	3	2	unsat	very_high	considering	LEAVE
4	one	29215	208	85	224784	241	21	1	very_unsat	little	never_thought	STAY

```
# Transform COLLEGE column to a numeric variable
#df["COLLEGE2"] = (df.COLLEGE == "one").astype(int)
df["COLLEGE2"] = (df.COLLEGE == "one").astype(int)
#df = df.drop("COLLEGE", axis="columns")
#df = df.drop("COLLEGE", axis="columns")
df.head(5)
```

Out[79]:

	COLLEGE	INCOME	OVERAGE	LEFTOVER	HOUSE	HANDSET_PRICE	OVER_15MINS_CALLS_PER_MONTH	AVERAGE_CALL_DURATION	REPORTED_SATISFACTION	REPORTED_USAGE_LEVEL	CONSIDERING_CHANGE_OF_PLAN	LEAVE	COLLEGE2
0	zero	31953	0	6	313378	161	0	4	unsat	little	no	STAY	0
1	one	36147	0	13	800586	244	0	6	unsat	little	considering	STAY	1
2	one	27273	230	0	305049	201	16	15	unsat	very_little	perhaps	STAY	1
3	zero	120070	38	33	788235	780	3	2	unsat	very_high	considering	LEAVE	0
4	one	29215	208	85	224784	241	21	1	very_unsat	little	never_thought	STAY	1

```
#df.REPORTED_SATISFACTION = df.REPORTED_SATISFACTION.astype('category')
#df.REPORTED_USAGE_LEVEL = df.REPORTED_USAGE_LEVEL.astype('category')
#df.CONSIDERING_CHANGE_OF_PLAN = df.CONSIDERING_CHANGE_OF_PLAN.astype('category')
#df.COLLEGE2 = df.COLLEGE2.astype('category')
###
df.REPORTED_SATISFACTION = df.REPORTED_SATISFACTION.astype('str')
df.REPORTED_USAGE_LEVEL = df.REPORTED_USAGE_LEVEL.astype('str')
df.CONSIDERING_CHANGE_OF_PLAN = df.CONSIDERING_CHANGE_OF_PLAN.astype('str')
```

```
df["LEAVE2"] = (df.LEAVE == "STAY").astype(int)
#df = df.drop("LEAVE", axis="columns")
df.head(5)
```

Out[81]:

	COLLEGE	INCOME	OVERAGE	LEFTOVER	HOUSE	HANDSET_PRICE	OVER_15MINS_CALLS_PER_MONTH	AVERAGE_CALL_DURATION	REPORTED_SATISFACTION	REPORTED_USAGE_LEVEL	CONSIDERING_CHANGE_OF_PLAN	LEAVE	COLLEGE2	LEAVE2
0	zero	31953	0	6	313378	161	0	4	unsat	little	no	STAY	0	1
1	one	36147	0	13	800586	244	0	6	unsat	little	considering	STAY	1	1
2	one	27273	230	0	305049	201	16	15	unsat	very_little	perhaps	STAY	1	1
3	zero	120070	38	33	788235	780	3	2	unsat	very_high	considering	LEAVE	0	0
4	one	29215	208	85	224784	241	21	1	very_unsat	little	never_thought	STAY	1	1

```
df.dtypes
```

```
COLLEGE                        object
INCOME                          int64
OVERAGE                         int64
LEFTOVER                        int64
HOUSE                           int64
HANDSET_PRICE                   int64
OVER_15MINS_CALLS_PER_MONTH     int64
AVERAGE_CALL_DURATION           int64
REPORTED_SATISFACTION          object
REPORTED_USAGE_LEVEL           object
CONSIDERING_CHANGE_OF_PLAN     object
LEAVE                          object
COLLEGE2                        int64
LEAVE2                          int64
dtype: object
```

```
predictor_cols = ["INCOME", "OVERAGE","LEFTOVER","HOUSE","OVER_15MINS_CALLS_PER_MONTH","AVERAGE_CALL_DURATION","COLLEGE2"]
target_col = "LEAVE2"

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df[predictor_cols],df[target_col],test_size = 0.5,random_state = 0)
```

```
print("X_train shape: {}".format(X_train.shape))
print("X_test shape: {}".format(X_test.shape))
print("y_train shape: {}".format(y_train.shape))
print("y_test shape: {}".format(y_test.shape))
```

```
X_train shape: (10000, 7)
X_test shape: (10000, 7)
y_train shape: (10000,)
y_test shape: (10000,)
```

```
from sklearn.tree import DecisionTreeClassifier
# Let's define the model (tree)
decision_tree = DecisionTreeClassifier(max_depth=6, criterion="entropy",max_leaf_nodes = 12, min_samples_leaf = 1)
# Let's tell the model what is the data
decision_tree.fit(X_train, y_train)
```

```
▸ DecisionTreeClassifier
```

```
# Using Decision Tree Classifier

from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier(random_state=0, max_depth=2)
scores2 = cross_val_score(decision_tree, X_test, y_test, cv = 10)
print("Cross validation scores: {}".format(scores2))
print(f'Average score: {"{:.2f}".format(scores2.mean())}')
print("Average accuracy for Trees: {:.2f}%".format(np.mean(scores2) * 100))
```

```
Cross validation scores: [0.657 0.64  0.644 0.672 0.676 0.63  0.662 0.648 0.646 0.658]
Average score: 0.65
Average accuracy for Trees: 65.33%
```

```
print("Average cross validation score: {:.2f}".format(scores2.mean()))
print("Std. dev of cross validation score: {:.2f}".format(scores2.std()*100))
print("Minimum of cross validation score: {:.2f}".format(scores2.min()))
```

```
Average cross validation score: 0.65
Std. dev of cross validation score: 1.37
Minimum of cross validation score: 0.63
```

Appendix A.3: Logistic regression cross-validation notebook: utilize the same notebook.

```
# Using Logistic Regression

from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
scores1 = cross_val_score(logreg, X_test, y_test, cv = 10)
print("Cross validation scores: {}".format(scores1))
print(f'Average score: {"{:.2f}".format(scores1.mean())}')
print("Average accuracy for Logistic Regression: {:.2f}%".format(np.mean(scores1) * 100))
```

```
Cross validation scores: [0.644 0.641 0.635 0.637 0.653 0.606 0.622 0.613 0.615 0.618]
Average score: 0.63
Average accuracy for Logistic Regression: 62.84%
```

```
print("Average cross validation score: {:.2f}".format(scores1.mean()))
print("Std. dev of cross validation score: {:.2f}".format(scores1.std()*100))
print("Minimum of cross validation score: {:.2f}".format(scores1.min()))
```

```
Average cross validation score: 0.63
Std. dev of cross validation score: 1.48
Minimum of cross validation score: 0.61
```

Part B:

Part B.1: The coefficients (BETAs) for the logistic regression model:

LR coefficients	Value
BETA0 (or constant term)	-2.0492
BETA1 (coeff. For X1)	0.3568
BETA2 (coeff. For X2)	1.1227

Appendix B.1: Logistic regression notebook:

```
In [...  import os
         import numpy as np
         import pandas as pd
         import math
         import matplotlib.pylab as plt
         import seaborn as sns

         %matplotlib inline
         sns.set(style='ticks', palette='Set2')

In [...  # Import data into Pandas as a data frame
         df = pd.read_excel('./Simmons-data-raw.xlsx')
         df.head(5)
```

Out[27]:

	Customer	Spending	Card	Coupon-Usage-Indicator
0	1	2.291	1	0
1	2	3.215	1	0
2	3	2.135	1	0
3	4	3.924	0	0
4	5	2.528	1	0

```
In [...  df.dtypes

Out[28]: Customer                   int64
         Spending                 float64
         Card                       int64
         Coupon-Usage-Indicator     int64
         dtype: object

In [...  df["Spending"] = (df.Spending).astype(int)

In [...  df.dtypes

Out[30]: Customer                 int64
         Spending                 int64
         Card                     int64
         Coupon-Usage-Indicator   int64
         dtype: object

In [...  import statsmodels.api as sm

         # Fit the logistic regression model
         model = sm.Logit(df["Coupon-Usage-Indicator"], sm.add_constant(df[["Spending", "Card"]])).fit()

         # View the model coefficients (Betas)
         print(model.params)

Optimization terminated successfully.
         Current function value: 0.603341
         Iterations 5
const      -2.049216
Spending    0.356808
Card        1.122659
dtype: float64

In [...  print(model.summary())

                           Logit Regression Results
==============================================================================
Dep. Variable:     Coupon-Usage-Indicator   No. Observations:               100
Model:                              Logit   Df Residuals:                    97
Method:                               MLE   Df Model:                         2
Date:                    Mon, 27 Mar 2023   Pseudo R-squ.:               0.1035
Time:                            02:54:22   Log-Likelihood:             -60.334
converged:                           True   LL-Null:                    -67.301
Covariance Type:                nonrobust   LLR p-value:              0.0009424
============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------
const         -2.0492      0.542     -3.782      0.000      -3.111      -0.987
Spending       0.3568      0.132      2.696      0.007       0.097       0.616
Card           1.1227      0.447      2.513      0.012       0.247       1.998
============================================================================
```

Part B.2:

	Probability of Response
Jack	0.48260373
Jill	0.39547087

Jack is likelier to respond because the response probability is higher than Jill's.

Appendix B.2: Predict customer notebook: utilize the same notebook.

```
In [...  predictor_cols = ['Spending','Card']
         target_col = ['Coupon-Usage-Indicator']

In [...  from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(df[predictor_cols],df[target_col],test_size = 0.5,random_state = 0)

         from sklearn import linear_model
         lin_model = linear_model.LogisticRegression(C = 1000, max_iter = 2000)

         # Fit the Logistic regression model now
         lin_model.fit(X_train, y_test)

         print(lin_model)

LogisticRegression(C=1000, max_iter=2000)
/Users/yangyang/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/validation.py:1111: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)

In [...  #test accuracy

         from sklearn import metrics

         print ("Accuracy = %.3f" % (metrics.accuracy_score(lin_model.predict(X_train), y_test)))

Accuracy = 0.620

In [...  y_pred = lin_model.predict(X_train)
         print("Test set score: {:2f}".format(np.mean(y_pred)))

Test set score: 0.140000

In [...  X_new1=np.array([[2,1]])

         def Predict_for_new_customer(X_new):
             pred_val = lin_model.predict(X_new)
             print("Prediction for new customer = ", pred_val)
             if(pred_val == 1):
                 return("Customer use coupon")
             elif(pred_val == 0):
                 return("Customer don't use coupon")
             else:
                 return("UNKNOWN STATUS")

         jack_prob = lin_model.predict_proba(X_new1)[:,1]
         print("Predicted value for Jack = " , Predict_for_new_customer(X_new1))
         print("Predicted probability of Jack = ", jack_prob)

Prediction for new customer =  [0]
Predicted value for Jack =  Customer don't use coupon
Predicted probability of Jack =  [0.48260373]
/Users/yangyang/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
/Users/yangyang/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(

In [...  X_new2 = np.array([[4,0]])
         jill_prob = lin_model.predict_proba(X_new2)[:,1]

         print("Prediction for Jill = ", Predict_for_new_customer(X_new2))
         print("Predicted probability of Jill = ", jill_prob)

Prediction for new customer =  [0]
Prediction for Jill =  Customer don't use coupon
Predicted probability of Jill =  [0.39547087]
/Users/yangyang/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
/Users/yangyang/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(

In [...  # print the results
         if jack_prob > jill_prob:
             print("Jack is more likely to use the coupon")
         elif jack_prob < jill_prob:
             print("Jill is more likely to use the coupon")
         else:
             print("Jack and Jill are equally likely to use the coupon")

Jack is more likely to use the coupon
```

Part B.3:

Before deciding the cut-off probability, the manager should understand a confusion matrix showing the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) at each threshold; after that, visualize the model's performance at different probability thresholds.

To make it easier to understand, we utilize the current dataset of 50 customers and create confusion matrixes based on each threshold, ranging from 0.1 to 1.0, as an example:

- From probability 0.1 to 0.3, the model falsely predicts that 28 customers, as FP, will use coupons when they will not, but no customers fall into FN or TN type.
- In probability 0.4, the situation changes when the model falsely predicts 25 customers as FP and 6 as FN, whom the model predicts will not use a coupon when they could, while correctly predicting 16 customers as TP and 3 as TN.
- In probability 0.5, the prediction becomes more negative with only 1 customer as TP and 25 as TN, while 21 customers as FN and 3 as FP.
- From probability 0.6 onward, it is opposite from 0.1 to 0.3, where 22 customers are FN, and 28 are TN.

When choosing a cutoff probability for logistic regression to predict coupon usage for an extensive database of customers, it's essential to consider the choice of the cutoff probability, which depends on the specific goals and constraints of the problem. For example, suppose the manager wants to maximize the number of customers who use coupons. In that case, choosing a lower-cutoff probability than 0.3 would be ideal. On the other hand, if the manager wants to minimize the cost of offering coupons to customers who do not use them, choosing a cutoff probability higher than 0.6 is the ultimate solution. From another perspective, if the manager focuses on the accuracy point of the model where the revenue can still be gained, probability 0.5 should be chosen since its accuracy is higher than the range 0.1 to 0.4 and gain 1 customer as TP, whereas the range of 0.6 to 1.0 could not gain any. For the probability 0.4, it could be an ideal solution for the revenue-cost balance where the company could still achieve 16 customers as TP and save the budget with 3 customers as TN.

In summary, the choice of cutoff probability depends on the specific requirements and objectives of the project, as well as the tradeoffs between false positives and false negatives. It is essential to evaluate the model's performance using different cutoff probabilities and select the one that best meets the project's requirements.

Appendix B.3: Confusion matrix notebook: utilize the same notebook.

```
unit_cost = 5
unit_revenue = 20
cost_matrix = pd.DataFrame([[unit_revenue - unit_cost, - unit_cost], [0, 0]], columns=['use coupon', 'not use'], index=['Y', 'N'])
print ("Cost matrix")
print (cost_matrix)
```

```
Cost matrix
   use coupon  not use
Y          15       -5
N           0        0
```

```
In [...
problist = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
probabilities = lin_model.predict_proba(X_test)[:, 1]

for i in problist:
    prediction = probabilities > i
    confusion_matrix = pd.DataFrame(metrics.confusion_matrix(y_test, prediction, labels=[1, 0]).T,
                                    columns=['use coupon', 'not use'], index=['Y', 'N'])
    print ("Confusion matrix with threshold", i," to predict labels")
    print (confusion_matrix)

    profit = np.sum((confusion_matrix * cost_matrix).values)
    print ("Expected profit per targeted individual with a cutoff of", i, "is $%.2f." % profit)
    print ("\n")
```

```
Confusion matrix with threshold 0.1  to predict labels
   use coupon  not use
Y          22       28
N           0        0
Expected profit per targeted individual with a cutoff of 0.1 is $190.00.


Confusion matrix with threshold 0.2  to predict labels
   use coupon  not use
Y          22       28
N           0        0
Expected profit per targeted individual with a cutoff of 0.2 is $190.00.


Confusion matrix with threshold 0.3  to predict labels
   use coupon  not use
Y          22       28
N           0        0
Expected profit per targeted individual with a cutoff of 0.3 is $190.00.


Confusion matrix with threshold 0.4  to predict labels
   use coupon  not use
Y          16       25
N           6        3
Expected profit per targeted individual with a cutoff of 0.4 is $115.00.


Confusion matrix with threshold 0.5  to predict labels
   use coupon  not use
Y           1        3
N          21       25
Expected profit per targeted individual with a cutoff of 0.5 is $0.00.


Confusion matrix with threshold 0.6  to predict labels
   use coupon  not use
Y           0        0
N          22       28
Expected profit per targeted individual with a cutoff of 0.6 is $0.00.


Confusion matrix with threshold 0.7  to predict labels
   use coupon  not use
Y           0        0
N          22       28
Expected profit per targeted individual with a cutoff of 0.7 is $0.00.


Confusion matrix with threshold 0.8  to predict labels
   use coupon  not use
Y           0        0
N          22       28
Expected profit per targeted individual with a cutoff of 0.8 is $0.00.


Confusion matrix with threshold 0.9  to predict labels
   use coupon  not use
Y           0        0
N          22       28
Expected profit per targeted individual with a cutoff of 0.9 is $0.00.


Confusion matrix with threshold 1.0  to predict labels
   use coupon  not use
Y           0        0
N          22       28
Expected profit per targeted individual with a cutoff of 1.0 is $0.00.
```