Name: **Shivam**
Email: **shivam@oregonstate.edu**
Course: **Machine Learning Challenges (Winter 2023)**
Date: **March 21, 2023**
Assignment: **Final Report**

1. The prediction problem I seek to address is being able to predict which customers are likely to stop using the services of a telecom company given their demographics and usage of the different services being offered by that company. When a customer stops using the services of a company (i.e. they cancel their subscription), this is known as churn. So, the aim of this project is churn prediction.
The real (or hypothetical) beneficiaries/users are any modern telecom companies interested in being able to predict which customers they are about to lose.

2. The dataset is provided/authored by IBM.
Quoting from their website: “The Telco customer churn data contains information about a fictional telco company that provided home phone and Internet services to 7043 customers in California in Q3. It indicates which customers have left, stayed, or signed up for their service.”
The detailed description of this dataset can be found at:
https://community.ibm.com/community/user/businessanalytics/blogs/steven-macko/2019/07/11/telco-customer-churn-1113
The URL for downloading this dataset is:
https://community.ibm.com/accelerators/catalog/content/Telco-customer-churn

Dataset profile:
There are a total of 7043 entries (rows) in the dataset.
There are a total of 33 columns in the dataset
The number of non-null items in each column, along with its data type is provided below:

RangeIndex: 7043 entries, 0 to 7042
Data columns (total 33 columns):

#	Column	Non-Null Count	Dtype
0	CustomerID	7043 non-null	object
1	Count	7043 non-null	int64
2	Country	7043 non-null	object
3	State	7043 non-null	object
4	City	7043 non-null	object
5	Zip Code	7043 non-null	int64
6	Lat Long	7043 non-null	object
7	Latitude	7043 non-null	float64
8	Longitude	7043 non-null	float64

```
 9   Gender            7043 non-null   object
 10  Senior Citizen    7043 non-null   object
 11  Partner           7043 non-null   object
 12  Dependents        7043 non-null   object
 13  Tenure Months     7043 non-null   int64
 14  Phone Service     7043 non-null   object
 15  Multiple Lines    7043 non-null   object
 16  Internet Service  7043 non-null   object
 17  Online Security   7043 non-null   object
 18  Online Backup     7043 non-null   object
 19  Device Protection 7043 non-null   object
 20  Tech Support      7043 non-null   object
 21  Streaming TV      7043 non-null   object
 22  Streaming Movies  7043 non-null   object
 23  Contract          7043 non-null   object
 24  Paperless Billing 7043 non-null   object
 25  Payment Method    7043 non-null   object
 26  Monthly Charges   7043 non-null   float64
 27  Total Charges     7032 non-null   float64
 28  Churn Label       7043 non-null   object
 29  Churn Value       7043 non-null   int64
 30  Churn Score       7043 non-null   int64
 31  CLTV              7043 non-null   int64
 32  Churn Reason      1869 non-null   object
dtypes: float64(4), int64(6), object(23)
```

It is important to note that of these 33 columns, IBM seems has added two additional columns based on the predictive modelling performed by one of their tools. These two columns are: “Churn Score” and “CLTV”. Because our aim is also to perform predictive analytics, hence, we will need to drop these two columns before doing any more work on this dataset. Similarly, the “Count” column, as explained above, is useful only for use with IBM’s internal tools. Thus, we will ignore it too.

“Churn Label” column is the same as the “Churn Value” column, except that it has Yes in place of 1 and No in place of 0 to represent whether the customer left (churned) the company or not, respectively. We will use the “Churn Value” column because our model deals with numbers.

Also, “Churn Reason” does not provide any useful information to the model to be able to predict whether a customer will churn or not because it is only available for customers who have already churned! However, for exploratory data analysis, that is, to understand why customers are churning and what factors are contributing to customer churn, then the "Churn Reason" column may be a valuable source of information. But this isn’t what we’re trying to do here.

And, the values for “Country”, “State” columns are the same for every customer. Moreover, the unique “Customer ID” column too isn’t useful as a feature for a machine learning classifier because the

identifier does not provide any predictive information about the target variable. Thus, these 4 columns will not be used as well.

Finally, I identified an additional fourth challenge in the dataset and that is to correctly represent the Latitude and Longitude columns as features to a machine learning classifier. I have considered two strategies (apart from the strategy of excluding these columns) that may be used for this purpose, but for this report, I do not yet have the numeric results and discussion for this additional challenge yet. Thus, I will leave a blank table for this challenge in this draft, and won't use the City, Zip Code, Latitude, Longitude, Lat Long columns in my analysis.
It is important to note that for the final version of the report too, I will not use the City, Zip Code and Lat Long columns because they will only serve to increase the dimensionality of the features if I use them. Instead, I will only use the two Latitude and Longitude columns to incorporate location-related features in the dataset as they provide far more information related to location than the other 3 columns.

Now, for the 20 columns that I am using, the number of items for each category for them are as follows (condensed as per feedback):

Gender: Male 3555, Female 3488
Senior Citizen: No 5901, Yes 1142
Partner: No 3641, Yes 3402
Dependents: No 5416, Yes 1627
Phone Service: Yes 6361, No 682
Multiple Lines: No 3390, Yes 2971, No phone service 682
Internet Service: Fiber optic 3096, DSL 2421, No 1526
Online Security: No 3498, Yes 2019, No internet service 1526
Online Backup: No 3088, Yes 2429, No internet service 1526
Device Protection: No 3095, Yes 2422, No internet service 1526
Tech Support: No 3473, Yes 2044, No internet service 1526
Streaming TV: No 2810, Yes 2707, No internet service 1526
Streaming Movies: No 2785, Yes 2732, No internet service 1526
Contract: Month-to-month 3875, Two year 1695, One year 1473
Paperless Billing: Yes 4171, No 2872
Payment Method:Electronic check 2365, Mailed check 1612, Bank transfer (automatic) 1544, Credit card (automatic) 1522
Churn Value (this is the label we want to predict): "0" 5174, "1" 1869

For the 3 numerical features, the min, max, mean, and median values are (put in 1 table as per feedback):

Feature Name	Minimum value	Maximum value	Mean value	Median value
Tenure Months	0	72	32.37	29.0
Monthly Charges	18.25	118.75	64.76	70.35
Total Charges	18.8	8684.8	2283.30	1397.48

3. The machine learning classifier I have used is the K Nearest Neighbors classifier, with K = 3. I chose this classifier because after doing some online search, I found the following advantages of a KNN classifier for the churn prediction problem:

(i) Because KNN is a non-parametric classifier, this implies that it doesn't make any assumptions about the distribution of the data. This makes it a robust classifier that can handle a wide range of data distributions.
(ii) Compared to SVM (which I initally intended to use), KNN is a simple and easy-to-understand algorithm. It is based on the intuitive idea of finding the closest neighbors to a new data point and classifying it based on the majority class of those neighbors.

Some strengths and weaknesses of KNN classifier are:
Strengths:
(a) It can work well for small datasets.
(b) It can be effective when the decision boundary is irregular.
Weaknesses:
(a) It can be sensitive to irrelevant features.
(b) It is likely to be biased towards the majority class in imbalanced datasets.

The following hyperparameters for a KNN classifier can be specified at the time of initialization (the values I used are given after the colon) -

{'algorithm': 'auto',
'leaf_size': 30,
'metric': 'minkowski',
'metric_params': None,
'n_jobs': None,
'n_neighbors': 3,
'p': 2,
'weights': 'uniform'}

Initially I have used the default values except for the n_neighbors value (which I set to 3).

Then, once I got the best strategies for each challenge, I used scikit-learn’s GridSearchCV(), which is a nice tool for hyperparameter tuning in machine learning models. It automates the process of finding the best combination of hyperparameters by systematically searching over a range of hyperparameter values and evaluating the model's performance on a validation set for each combination. The code for this task is provided in the fputils.py file.

4. I used the Stratified10-Fold Cross validation strategy.
The baseline algorithm I used is the Dummy Classifier from scikit-learn, with strategy=‘stratified’ . Apart from accuracy, I used the precision and recall metrics, which are helpful for imbalanced datasets. Also, for this classification problem, minimizing false negatives (customer actually left the company but classifier predicted the customer as retained by the company) was of paramount importance. Thus, I also used the F2 score metric.

5. After another review of the dataset, I found 4 major challenges in this dataset.

The fourth challenge that I talked about in my initial findings report is a minor challenge, and as instructed by the professor, I have performed an initial pre-processing to address this minor challenge. The minor challenge was about handling missing values. There were only 11 missing values from the Total Charges column. These values are missing only (MAR) for those customers which have joined the company and remained as customers for less than a month. That is, their corresponding value for the column Tenure Months = 0. I replaced the missing values in the Total Charges column with the values in the Monthly Charges column (which are not missing for any row). Thus, for customers with Tenure = 0 (month), there Total Charges are now the same as Monthly Charges.

The final 4 challenges in this dataset are:

(I) The first challenge in my dataset is the handling of categorical features. For the chosen dataset, the following columns contain categorical data: Gender, Senior Citizen, Partner, Dependents, Phone Service, Multiple Lines, Internet Service, Online Security, Online Backup, Device Protection, Tech Support, Streaming TV, Streaming Movies, Contract, Paperless Billing, Payment Method.
Thus, we need to transform (or encode) this data into numeric data before it can be utilized by a model. The three ideas I have used are:
(a) The first idea for this challenge is to use one-hot encoding. One-hot encoding for categorical features, produces one feature per category, each binary.
(https://contrib.scikit-learn.org/category_encoders/onehot.html)
(b) The second idea for this challenge is to use ordinal encoding. It encodes categorical features as ordinal, in one ordered feature. Ordinal encoding uses a single column of integers to represent the feature values. An optional mapping dict can be passed in; in this case, we use the knowledge that there is some true order to the classes themselves. Otherwise, the feature values are assumed to have no true order and integers are selected at random.
(https://contrib.scikit-learn.org/category_encoders/ordinal.html)
Now, ordinal encoding only makes sense if there is a rank ordering of the feature values. This is not possible for all the features in the dataset. Hence, I have used it for the ones that have a rank ordering, and one-hot encoding for the other remaining categorical features.
(c) The third idea for this challenge is to use count encoding. For a given categorical feature, it replaces the names of the groups with the group counts.
(https://contrib.scikit-learn.org/category_encoders/count.html)

(II) The second challenge in my dataset is adjusting the range for certain features, i.e., performing feature scaling. For the chosen dataset, the minimum and maximum values for the Monthly Charges column are 18.25 and 118.75, respectively. Similarly, for the Total Charges column the minimum and maximum values are 18.8 (ignoring missing values for now) and 8684.8, respectively. Finally, for the Tenure Months column, the minimum and maximum values are 0 and 72, respectively. Clearly, the range of values of data in these three columns varies widely.
So I have used the following three approaches for feature scaling:
(a) The first idea is to use the Min Max Scaler. It transforms features by scaling each feature to a given

range. This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g. between zero and one.
(https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html)
(b) The second idea is to use the Standard Scaler. It standardizes features by removing the mean and scaling to unit variance.
(https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html)
(c) The third idea is to use the Robust Scaler. It scales features using statistics that are robust to outliers.
(https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html)

(III) The third challenge I want to address in my dataset is handling class imbalance. For the chosen dataset, Churn Value is our label column. Out of the total 7043 rows, there are 1869 rows of customers who left the telecom company (Churn Value = 1) and 5174 rows of customers who remained with the telecom company (Churn Value = 0). Clearly, there is significant class imbalance in this dataset.
The three ideas I intend to use for this challenge are:
(a) The first idea is to simply do nothing and use the same imbalanced dataset for training the classifier.

(b) The second idea is to use the oversampling technique called random oversampling.
(https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.RandomOverSampler.html)
For this strategy, the minority class (Churn Value = 1) has been oversampled by 3305 extra samples.

(c) The third idea is to use the undersampling technique called random undersampling.
(https://imbalanced-learn.org/stable/references/generated/imblearn.under_sampling.RandomUnderSampler.html)
For this strategy, the majority class (Churn Value = 0) has been undersampled by 3305 less samples.

(IV) The fourth challenge as described earlier, is about representing latitudes and longitudes (location-related information) appropriately so they can be used as features for a machine learning classifier.
I have tried three approaches:

(a) The first approach is to exclude the Latitude and Longitude columns from the dataset. It is noteworthy that the original version of this dataset did not contain any location related information. IBM revised it and added more columns to reflect location-related information in the dataset.

(b) The second approach is what I found on the StackExchange answer below:
https://datascience.stackexchange.com/questions/13567/ways-to-deal-with-longitude-latitude-feature
Basically, the latitude and longitude values are first mapped to x, y, and z coordinates using the equations x = cos(lat) * cos(lon), y = cos(lat) * sin(lon), and z = sin(lat). The resulting x, y, and z coordinates are then combined into a new DataFrame (new_df).

Next, the StandardScaler class from Scikit-learn can be used to standardize the x, y, and z coordinates in new_df. The standardized values are then used to replace the original Latitude and Longitude columns in the original DataFrame telco_df.

(c) The third approach is to use the latitude and longitude column values to calculate the great-circle distance using the Haversine formula. In order to calculate the great-circle distance between two locations, the first location is a fixed location and it is a major city in the state of California, i.e., Los Angeles. The formula is given at: https://www.igismap.com/haversine-formula-calculate-geographic-distance-earth/

6.
The baseline classifier's (for all challenges) accuracy, recall, precision, and F2 scores, respectively are as follows:
Baseline accuracy: 61.39
Baseline precision: 25.15
Baseline recall: 23.01
Baseline F2 score: 23.40836097403123

I am also putting the Baseline results at the bottom row of each table, as per the feedback.

The experimental results for KNN classifier for challenge 1 (handling of categorical features) are as follows:

Strategies	Accuracy	Recall	Precision	F2 Score
One-Hot Encoding	75.18	46.82	53.79	48.07
Ordinal Encoding with One-Hot Encoding	75.17	46.76	53.77	48.01
Count Encoding	76.53	50.72	56.46	51.77
Baseline classifier	61.39	23.01	25.15	23.41

Discussion for challenge 1 (handling of categorical features):
It is clear that the count encoding (aka frequency encoding) scheme worked better than the other two strategies which were almost similar. The accuracy is almost same, but the F2 score of count encoding is considerably better than the other two encodings.
Now, it is possible that One-Hot encoding and Ordinal encoding may have increased the dimensionality of the feature space. This increase in dimensionality could have lead to overfitting of the model. Count

encoding, on the other hand, does not add new dimensions to the feature space (as can be verified from nparray.shape) and can provide a tighter representation of the categorical variables.
Moreover, the frequency of occurrence of each category may have been more informative for some categorical variables than others, which is another likely reason that made count encoding's performance better than the other two.

The experimental results for for KNN classifier for challenge 2 (performing feature scaling) are as follows:

NOTE: Because some kind of encoding for categorical data had to be performed, hence, for the challenges below, I used the count encoding strategy which was the best among the three strategies above.

Strategies	Accuracy	Recall	Precision	F2 Score
No scaling	76.53	50.72	56.46	51.77
MinMax Scaler	75.56	49.12	54.41	50.09
Standard Scaler	75.58	49.06	54.45	50.05
Robust Scaler	75.55	49.01	54.39	50.00
Baseline classifier	61.39	23.01	25.15	23.41

Discussion for challenge 2 (performing feature scaling):
It is evident that there is practically no difference between the different scaling strategies. In fact, as mentioned in the feedback, indeed, the accuracy scores did drop after scaling. One possible explanation is that the data is already normalized or standardized and doesn't really require scaling. Another possible explanation is that for this particular dataset, the difference between the scaling methods used is not significant enough to produce noticeable differences in the results. Additionally, the scaling of some features may have caused underfitting in the model. If the scaling was too strong and since the number of neighbors in KNN is too small (I set it at 3 initially), the model may not have been able to capture the patterns in the data, leading to poor performance. Finally, because the three different scalers performed almost similarly, it also means that there isn't much noise in the data nor are there any outliers.

The experimental results for KNN classifier for challenge 3 (handling class imbalance) are as follows:

Strategies	Accuracy	Recall	Precision	F2 Score
Do nothing	76.53	50.72	56.46	51.77
Random Oversampling	81.38	89.89	76.82	86.93
Random Undersampling	71.72	73.2	71.15	72.78
Baseline classifier	61.39	23.01	25.15	23.41

Discussion for challenge 3 (handling class imbalance):
A simple reasoning based explanation for the above results is that in the original dataset, the minority class was significantly underrepresented (there was significant class imbalance), which led to the model being biased towards the majority class. By randomly oversampling the minority class, the size of the minority class was increased, and the model was trained on a more balanced dataset. This allowed the model to learn more features and patterns from the minority class and make better predictions on the test data. On the other hand, random undersampling reduced the size of the majority class, which possibly led to loss of important information and patterns in the data, thus reducing the performance of the model, even compared to the “do nothing” strategy.

The experimental results for KNN classifier for challenge 4 (representing latitudes and longitudes, i.e., location-related information appropriately) are as follows:

Strategies	Accuracy	Recall	Precision	F2 Score
Exclude the location-related features	76.53	50.72	56.46	51.77
Represent them as x,y,z values and then standardize them	76.47	50.4	56.37	51.49
Use the Haversine Distance	75.31	47.78	53.95	48.90
Baseline classifier	61.39	23.01	25.15	23.41

Name: Shivam Assignment: Final Report

Discussion for challenge 4 (representing latitudes and longitudes, i.e., location-related information appropriately):
It turns out that not including the Latitude and Longitude columns was the best choice. There might have been several reasons for this. First, the latitude and longitude information was not adding any unique information or value to the classifier, hence removing the column was the best choice. Second, converting the latitude and longitude information into x,y,z coordinates or Haversine distances could have added complexity to the model, which may not have been necessary or useful. Third, if the latitude and longitude information was missing or inaccurate for some of the customers, then the conversion methods may have produced unreliable or misleading results. Fourth, because high-dimensional data might have lead to overfitting and other modeling problems. Hence, removing the latitude and longitude column likely reduced the dimensionality of the dataset, which may have improved the performance of the classifier.

7. (a) I was surprised that the location related information didn't really add anything useful to the classifier. It took a lot of time to get the two strategies for handling latitudes and longitudes right, but the end result was that dropping these columns was the best choice indeed.

(b) One point of feedback was to explain what could be the possible reason that feature scaling isn't even effective for a classifier like KNN? Another was that compared to other transformations, such as scaling or encoding, random oversampling seems like a very simple solution. It was surprising that such a basic transformation could have such a dramatic impact on the model's performance. It would be interesting to understand better why this transformation works so well for this specific dataset and how it affects the KNN classifier's decision boundaries.

(c) Originally, I was including feature scaling in every subsequent challenge strategy (i.e. from challenge 2 onwards). But after receiving the feedback, I decided not to go ahead with feature scaling in the subsequent challenges as it wasn't really improving things.

(d) I would definitely investigate some other class imbalance strategies for handling the third challenge, i.e., the class imbalance challenge. In addition, I would look at how to determine the importance of each feature using a statistical test like ANOVA.

Finally, the best performance of the classifier is:
Average KNN accuracy: 86.1
Average KNN precision: 80.07
Average KNN recall: 96.17
KNN F2 score: 92.45205787009243