Part 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.

Parts 2 and 3:
1. The first challenge in my dataset is the handling of categorical features. For the chosen dataset, several columns like City, Zip Code, 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, Churn Reason contain categorical data. The list of categories for each column can be found in my project proposal document.

Thus, we need to transform (or encode) this data into numeric data before it can be utilized by a model. The three ideas I plan to use are:
(a) The first idea for this challenge is to use *one-hot encoding*. One-hot coding 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 classes. 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 classes are assumed to have no true order and integers are selected at random. (https://contrib.scikit-learn.org/category_encoders/ordinal.html)

(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)

2. The second challenge in my dataset is adjusting the range for certain features, i.e., performing feature scaling. For the chosen dataset, for example, 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. Clearly, the range of values of data in the two columns varies widely.

Because I intend to use the SVM classifier, and thanks to these StackExchange answers:
(i) (https://stats.stackexchange.com/questions/244507/what-algorithms-need-feature-scaling-beside-from-svm)
(ii) (https://stackoverflow.com/questions/26225344/why-feature-scaling-in-svm),
I found out that feature scaling will be useful for me.

So I will use 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)

3. 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)

(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)

4. The fourth challenge, which is probably a minor challenge compared to the other three, is handling missing values. It is worth mentioning that there are only 11 missing values for the Total Charges column. These values are missing only (MAR) for those customers which left the company in less than a month (Tenure = 0). Instead of trying out three ideas, I think a very reasonable way to handle this would be to replace 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 would be the same as Monthly Charges. This is what we observe for customers with Tenure = 1 (month) as well.

Part 4:
I intend to use *Kfold* Cross Validation as well as *StratifiedKFold* Cross validation. StratifiedKFold will be especially useful when the dataset is left imbalanced.

The baseline algorithm I intend to use is the *Dummy Classifier* from scikit-learn, with strategy = ‘uniform’ or ‘stratified’, depending on whether the dataset has been balanced or not.

Apart from accuracy, I intend to use 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) is of paramount importance. Thus, the *F2* score would also be a useful metric here.