**Project 2: Portuguese Banking Product**

**Group members :** Amol Madhukar Nanaware; Anurag Chaturvedi; Bindu Nag;

Luke Hannon; Frederick Alexander Pynn; Jyothi Sara Thomas;

Nishant Kumar

**All group members have read and agreed to the final version of all documents.**

**Signatures**

1. Amol Madhukar Nanaware (19250328)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. Anurag Chaturvedi (19251412)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
3. Bindu Nag (19251016)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_
4. Luke Hannon (19253525)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
5. Frederick Alexander Pynn (19251367)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
6. Jyothi Sara Thomas (19250822)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_
7. Nishant Kumar (19251001)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Introduction**

The following report pertains to data[1] collected from direct marketing campaigns of a Portuguese banking institution. The goal was to model the probability of successfully securing a product subscription (in this case, a bank term deposit). This marketing campaign was conducted via phone calls made to clients.

Two data files were used; bank-additional-full.csv (this is the full data set for training and testing) and bank-additional.csv (this is the testing dataset). Both datasets were downloaded from the UCI website[1]. There are total 41,188 observations and 21 variables in the dataset, of which 10 are continuous variables and 10 categorical variables. The target response (labelled y in the dataset) is binary, and indicates whether the client subscribed to a term deposit or not. ‘yes’ indicates the client subscribed to a term deposit, and ‘no’ indicates the client did not subscribe to a term deposit. The predictors are divided into 4 categories: client data, last contact of current campaign, campaign attributes, and social and economic attributes.

In this project’s analysis, logistic regression models were fit using three different languages; SAS, R, and Python. All the steps[4] to check the data, clean the data, and fit each model where done separately and in full in each language. The experience of each group tasked with performing these tasks in each language are detailed in the sections below. Regarding software use, the Python was written in Jupyter Notebook, the SAS in SAS ODA, and the R in RStudio.

# All Languages

Although the specifics of the coding and analysis varied among the languages, the high-level steps followed by each group were kept consistent. These steps were: importing, handling outliers, handling missing data, splitting the testing and training sets, basic model fitting, model refinement, and test scoring. An important note is that the terms testing, and validation will be treated as synonymous in this report, even though they are usually distinct datasets.

Outliers were found and removed from the duration and campaign columns. Also, any records containing an unknown result were deleted. Imputing these values was debated, but since all the unknown values were associated with categorical variables, it was deemed too difficult to accurately impute them. As mentioned in the project description, the *duration* variable was excluded from all models, as it could not be used to predict the success of a call before the call was actually made.

For choosing which models were “better”, it was decided that AIC would be used, as it is a simple and well-regarded metric for model comparison. In addition to the numeric quantifiers, ROC curves were included to aid in scoring the models. ROC (Receiver Operating Characteristic) curves are useful for models with binary output, as they give the proportion of observations that are correctly identified by the model. The area under the curve (AUC) was used as a metric of how well the model performed, along with the resulting confusion matrix.

# Python (Amol Nanaware & Nishant Kumar)

For Logistic Regression, python has two libraries(packages), *sklearn* and *StatsModels*. Both the packages have different ways of calculation and analysis. *StatsModels* package gives reports in more details as compare to *sklearn* by using `*summary2*` method on the model fit.

In our analysis we have used *sklearn* package to analyse the data and we are doing so based on three metrics, Area under the curve (AUC ROC score) and Mean accuracy of model and prediction accuracy. In python, we tried 4 ways of analysis, one with full model, second with reduced model (13 predictors), reduced model with 3 predictors, and reduced model with one predictor. The results can be seen in *suppl. table 1*.

So, the best model is with 13 variables in which we get AUC score of 0.8 which is highest with mean accuracy of 74.60%.

Model = month + education + contact + job + day\_of\_week + poutcome + marital + cons.conf.idx +cons.price.idx + emp.var.rate + previous + pdays + campaign

With this reduced model we got 82% accuracy on the test data(bank-additional.csv). To reduce the model, we have used correlation(*Fig.4* - highly correlated variables should be removed), backward elimination and Recursive feature elimination with cross-validation.

For encoding predictors, we have used `*get\_dummies*` method of *pandas* library, which creates separate columns for each unique value of categorical predictor.

Using ‘*corr’* function of pandas we have found around 5 highly correlated predictors and removed them. In 3 predictor model, predictors are selected based on high correlation of predictors with target variable(y). This means that these 3 variables highly influence the prediction of term deposit. These 3 predictors are ‘pdays’, ‘previous’ and ‘emp.var.rate’.

Finally, we tested the model with 1 highly influencing variable which is ‘pdays’ and the results can be seen in *suppl. table 1.*

# SAS (Luke Hannon, Anurag Chaturvedi & Alex Pynn)

First the bank data csv file was imported into SAS and checked for anomalies and outliers. The variables *nr.employed* and *Euribo3m* were found to be highly correlated with *emp.var.rate*. They were removed and *emp.var.rate* was kept. Using SMOTE (synthetic minority over-sampling technique), reducing the class imbalance in the *y* response variable was attempted. This technique was ultimately abandoned due to the difficulty of working with categorical variables. Once the cleaning was complete, the data was split into a training and testing set. Splitting the data was a slight challenge, as it was not immediately evident that the splitting had happened. Although looking around on the internet made it clear there were several ways to split data into testing and training sets, the method used here simply added a column with a categorical training (1) or testing (0) output. The data had to be manually split into two sets after this was done.

**Basic Model**

First, a simple logistic model was fit to the job predictor to ensure the command in SAS was being used properly. The command consisted of three primary components:

1. The Class Statement: This is required to allow SAS to recognize which variables are categorical and treat them appropriately in the model. This turned out to be a big time-saver, as there was no need to change any categorical variables to numeric ones.
2. The Model Statement: This defines the structure of the model itself. The model can also be labeled for easy identification in output comparison.
3. The Score Statement: This takes the testing data as input and is used to produce an AUC score that can be compared to the score produced by the training data.

By far the most difficult part of working with SAS was figuring out how to properly score the models and compare the training to the testing data. It was initially attempted to score the models in a statement separate from within the proc logistic statement. It was assumed that this would be straightforward but turned out to be very cumbersome. Eventually it was discovered that the scoring could be done along with the model fitting, so this was chosen as the simplest way to gauge the accuracy of the models.

**Full Model**

Once the simple model was run successfully, a full model was created to check the effect of all variables. Backwards elimination was used in the model statement to eliminate non-significant variables. Once the significant variables were identified, another model was fit with only these variables where they were analyzed with more scrutiny to determine how much they contributed to the model. An additional model was fit to see if condensing the job categories would improve the model, but this did not turn out to be effective.

# R (Bindu Nag & Jyothi Sara Thomas)

The bank-additional-full dataset is read, and R automatically parses all the string columns as factors. We have used the *read\_csv()* function to read the data as opposed to the other popular command *read.csv()* as the size of the dataset is considerably large. For missing values in the dataset, we replace the “unknown” values with NA, this makes it easier to handle and impute the missing columns. We also encoded all the categorical, string columns into numerical values. We removed the default column which had 8957 missing values. It is suggested to remove any missing columns with greater than 50% of missing values but here, we observe around 21% of the values missing for default which may pose an issue while fitting the model. We have split the cleaned dataset into test and train set in a ratio of 30:70.

**Initial Model**

We initially fitted the model with the removed default column, but R fails to fit the stepwise regression method if there is any NA in the dataset. To overcome this challenge, we processed data as below:

* We removed the rows with NA values loan and housing and imputed values for education using the mean. Using the imputed education column, we imputed the job column using the group median of education category.
* Encoded the categorical variables to dummy variables.

Upon fitting these models using stepwise regression, which is done by the *step()* function in the R package *stats* which selects a formula-based model by AIC. The results obtained for the split test data are seen in *suppl. table 2*.

**Final Model**

We also tested our fit on the dataset cleaned using Python (Encoded Dummy) as it produced the best results overall.

On fitting a logistic regression with the stepwise elimination method on this dataset we obtain an accuracy of 75.6%, the AUC score of 74.69% but an AIC of 39471.

y ~ age + housing + campaign + pdays + previous + emp.var.rate + cons.price.idx + cons.conf.idx + job + education + month + day\_of\_week +poutcome

Encoding categorical to dummy variables both in R and Python, we observe that though the data processing was similar, upon fitting the data yields different results. So, it is fair to assume R and Python fit the model differently.

It is up to the requirement of the business to conclude as to which model fitting is preferable and appropriate to their specification.

# Conclusion

Based on the compiled information, all three languages produced reasonably successful models. The researcher who originally analysed the dataset obtained an AUC of 0.803[2]; success rate that the SAS and Python models both approximate. It is interesting to note that SAS gives the best prediction accuracy, but in terms of independent model fitting then R gives highest accuracy, and yet has a low AUC score.

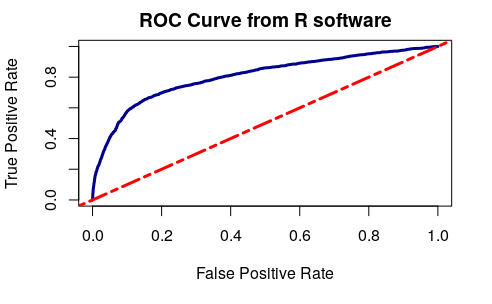
Regarding the ease of model construction and fitting, Python and R both proved to be easier to work with and more flexible. This, however, seems to come with a trade-off of performance, as the SAS model seems to, in general perform best. It’s also interesting to note the difference in execution time, as the SAS information must be sent, processed, and then received. This is less important for a small project like this, but could become a big time sink if a larger project was underway. Detailed results of the final models can be seen in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter\Software | SAS | Python | R |
| Number of predictors | 13 | 13 | 13 |
| Prediction accuracy of model | 88.9% | 81% | 75.6% |
| Model mean Standard Deviation | 6.1965 | 0.53 | 1.7253 |
| Mean AUC ROC Score | 79.4% | 79.75% | 74.69% |
| Model Execution time | 5.06 s | 669 ms | 0.689 s |
| Ease of coding (5 being the best) | 3/5 | 4/5 | 3.8/5 |
| Confusion matrix | |  |  |  | | --- | --- | --- | |  | Predicted 0 | Predicted 1 | | 0 | 7843 | 145 | | 1 | 870 | 287 |   Found method:  proc freq data=TestOutput1;  tables F\_y\*I\_y;  run; | |  |  |  | | --- | --- | --- | |  | Predicted 0 | Predicted 1 | | 0 | 6613 | 382 | | 1 | 1398 | 754 |   In-built method :  confusion\_matrix(  y\_test, y\_pred) | |  |  |  | | --- | --- | --- | |  | Predicted 0 | Predicted 1 | | 0 | 7038 | 1017 | | 1 | 3010 | 4913 |   Found method:  table(y\_test, y\_pred>0.5) |
| Clarity of output | Output is very clear, although there is a large quantity of information to sift through, so figuring out which information is relevant/important is somewhat difficult | Confusion matrix, accuracy is provided by the predict function but if we need more details we need to rely on cross-validation function and if we need even more details then we need to use *StatsModels* library and fit the model and use summary function. | R has function named *accuracy* from *SDMTools* packagethat provides tabular output of AUC score, prediction accuracy, sensitivity and other properties which makes the results clear and concise. |

# References

1. [Moro et al., 2014] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014 (<http://archive.ics.uci.edu/ml/datasets/Bank+Marketing>)
2. <https://github.com/sukanta-27/Predicting-Success-of-Bank-Telemarketing/blob/master/Relevent%20Paper/targeted_marketing.pdf>
3. Full project is uploaded on GitHub: <https://github.com/NanawareAmol/Project_Portuguese_Banking>
4. https://github.com/NanawareAmol/Project\_Portuguese\_Banking/blob/master/Steps.txt

# Supplementary Tables and Figures



*Figure 1: R model ROC Curve*

![A close up of a map

Description automatically generated](data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAYABgAAD/4RD0RXhpZgAATU0AKgAAAAgABAE7AAIAAAAOAAAISodpAAQAAAABAAAIWJydAAEAAAAcAAAQ0OocAAcAAAgMAAAAPgAAAAAc6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEFtb2wgTmFuYXdhcmUAAAWQAwACAAAAFAAAEKaQBAACAAAAFAAAELqSkQACAAAAAzkxAACSkgACAAAAAzkxAADqHAAHAAAIDAAACJoAAAAAHOoAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAyMDIwOjA0OjI5IDEyOjA5OjE5ADIwMjA6MDQ6MjkgMTI6MDk6MTkAAABBAG0AbwBsACAATgBhAG4AYQB3AGEAcgBlAAAA/+ELIGh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC8APD94cGFja2V0IGJlZ2luPSfvu78nIGlkPSdXNU0wTXBDZWhpSHpyZVN6TlRjemtjOWQnPz4NCjx4OnhtcG1ldGEgeG1sbnM6eD0iYWRvYmU6bnM6bWV0YS8iPjxyZGY6UkRGIHhtbG5zOnJkZj0iaHR0cDovL3d3dy53My5vcmcvMTk5OS8wMi8yMi1yZGYtc3ludGF4LW5zIyI+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczpkYz0iaHR0cDovL3B1cmwub3JnL2RjL2VsZW1lbnRzLzEuMS8iLz48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOnhtcD0iaHR0cDovL25zLmFkb2JlLmNvbS94YXAvMS4wLyI+PHhtcDpDcmVhdGVEYXRlPjIwMjAtMDQtMjlUMTI6MDk6MTkuOTEyPC94bXA6Q3JlYXRlRGF0ZT48L3JkZjpEZXNjcmlwdGlvbj48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOmRjPSJodHRwOi8vcHVybC5vcmcvZGMvZWxlbWVudHMvMS4xLyI+PGRjOmNyZWF0b3I+PHJkZjpTZXEgeG1sbnM6cmRmPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5LzAyLzIyLXJkZi1zeW50YXgtbnMjIj48cmRmOmxpPkFtb2wgTmFuYXdhcmU8L3JkZjpsaT48L3JkZjpTZXE+DQoJCQk8L2RjOmNyZWF0b3I+PC9yZGY6RGVzY3JpcHRpb24+PC9yZGY6UkRGPjwveDp4bXBtZXRhPg0KICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICA8P3hwYWNrZXQgZW5kPSd3Jz8+/9sAQwAHBQUGBQQHBgUGCAcHCAoRCwoJCQoVDxAMERgVGhkYFRgXGx4nIRsdJR0XGCIuIiUoKSssKxogLzMvKjInKisq/9sAQwEHCAgKCQoUCwsUKhwYHCoqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioq/8AAEQgBFwGaAwEiAAIRAQMRAf/EAB8AAAEFAQEBAQEBAAAAAAAAAAABAgMEBQYHCAkKC//EALUQAAIBAwMCBAMFBQQEAAABfQECAwAEEQUSITFBBhNRYQcicRQygZGhCCNCscEVUtHwJDNicoIJChYXGBkaJSYnKCkqNDU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6g4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS09TV1tfY2drh4uPk5ebn6Onq8fLz9PX29/j5+v/EAB8BAAMBAQEBAQEBAQEAAAAAAAABAgMEBQYHCAkKC//EALURAAIBAgQEAwQHBQQEAAECdwABAgMRBAUhMQYSQVEHYXETIjKBCBRCkaGxwQkjM1LwFWJy0QoWJDThJfEXGBkaJicoKSo1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoKDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uLj5OXm5+jp6vLz9PX29/j5+v/aAAwDAQACEQMRAD8A+kaKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACivHfBg8XT+HdL1nRYtUMv9kyy3L6zqRuodQmKAxeXH5zsmWHXEeAcYPStLxP4z1HXfDOpXHhW6js7S3tLGb7UUcyCWeRW8sFXXG2MqWH/TQD1oA9QorzPUp/EuneO9UuLO50qW8tfDcM9xLLaSLHLtmuSEVBKSmR/EWbGOhzxavfHery2GqatpEFjHYaLp0N/cwXSu0twHi85kRgwCYTADENluMDGaAPQqK8+1Dxnr9vfa9d28enf2Tot/aW7RSQyefPHLFA7fNvwjL5xI+U54GBjJ7TVNWt9HtVuLuO8kRnCAWllNctnBPKxKzAcdSMdOeRQBdorkfiTcXEPha1Nob3dNqllE0djcG3mlR7hFZFcMm0sCR95evUVycGt6va+HdXtorm/wBNWTxDa6bEmoXH2i702GXyVctIWcEsWYodzgb15OMAA9aorzPxNe3vw+v2Oh3l9fx3GjX901nf3cl0Vlt41dJFZyWAJbaVBwcjABrp/Dnh1LaCw1Ntb1W/uXiEk0s1/I8NyWTr5RPlqMnI2KuMDtnIB0tFc1quraxP4rXQdAextpI7IXs9zewPMCGcoqKiunOVYli3HHBzWdqXjLUbCw8VS+VaGTRb+2tYMq2HWSK3di3zcndMwGMdB15yAdtRXB3PibxSw8VXlhFprWmgXTRR2xt5HmulWGKVhuEgCth2A4OTjgY5ZqvxEnhXULnSYYrmxS4s7G0mWCWYvPMvmO5WPLOixvHhVGS2RkdgDv6K5zwfruoa1DfJqlrIj2s4SK5Onz2SXSFQ25YpvmXBJUjJHGc84HR0AFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFY3iHWrvRP7PkgsI7q3uLyG1ndrgxtD5sqRqyrtO/l84yvA61s1heKtF1HXbO1g03UbWx8m6huma4s2n3tFIsiAYkTAyvPXIPagCXXNautPurKx0uxjvr+88xkjmuDCioi5Zi4Vj1KgDHVh0GTVnSdXt9W8P2msR5ht7q2W4HmkAxqV3YbsMd/pVDVND1K+ms7221O2ttSs2kEcxs2eMxyKAysnmAk5VWB3DlRwec6OjaXFomh2Wl2zM8VnAkCM/wB5gqgZPucZoAzh468JEkDxTopIOD/xMIuP/HqX/hOfCf8A0NGi/wDgwi/+Ko8P/wDIc8Uf9hRP/SK2rdoAwv8AhOfCf/Q0aL/4MIv/AIqj/hOfCf8A0NGi/wDgwi/+KrdooAwv+E58J/8AQ0aL/wCDCL/4qj/hOfCf/Q0aL/4MIv8A4qt2igDC/wCE58J/9DRov/gwi/8AiqP+E58J/wDQ0aL/AODCL/4qt2igDC/4Tnwn/wBDRov/AIMIv/iqP+E58J/9DRov/gwi/wDiq3aKAML/AITnwn/0NGi/+DCL/wCKo/4Tnwn/ANDRov8A4MIv/iq3aKAML/hOfCf/AENGi/8Agwi/+Ko/4Tnwn/0NGi/+DCL/AOKrdooA5Sz8T+DtE0GPT9D1/QljtIPLtYJNWQLwPlUuSzAe+CfY1heHG8D6d4Pu9H1XxB4el/tK4mur6ODUkEe+Vy21DuDYUbVU8HCA8V6RRQBxkGo/D6DziPEWmSvPaCylkn1rzneEM7BSzyEnmR+c55xngYq3I+GN20Bn1jRyIII7cIusBUljj+4kiiQCUDsHDdTXe0UAcbcan8PLqLUo59d0V11SZJ7sf2og810VFU8PxgRIMDA49znV/wCE58J/9DRov/gwi/8Aiq3aKAOY1DxN4H1SGKK/8RaLKkM8dzGP7SjXEkbh0bhh0YA46HvVW71T4d3899Le61oU7ajAlvdq+oxlZkQsVyu/GRuOGxnpzwK7GigDidKvfh1o1xLcWev6U9xNGIXnu9Z+0yeX12BpZGIX/ZBx7VHpk3w30e8iuNO1/TIjASYITrheGDIIPlxNIUTgkfKo4Nd1RQBxur6p8P8AW54Z7/xHpYuIFZI57bWvs8gVsZXfHIrFTgZUnHHSqepL8MdWvJLm/wBa0mR5RH5qrrWxJDHjYzIsgVmXAwxBIx1rvqKAOXs/EvgewkvJLTxFoqNfTm4uD/aUbeZJsVM8tx8qKMDjis6yl+Gen+Gf+Efs9W0OPS9+8Qf2ohKtu3BgxfcCGAIIOVwMYwK7migDlNK8Q+CNGt3isvFOnsJG3s11rf2hycY+/JIzdumavf8ACc+E/wDoaNF/8GEX/wAVW7RQBhf8Jz4T/wCho0X/AMGEX/xVH/Cc+E/+ho0X/wAGEX/xVbtFAGF/wnPhP/oaNF/8GEX/AMVR/wAJz4T/AOho0X/wYRf/ABVbtQXt9aabaPdajdQ2lvGMvNPIERfqTwKaTk7IDJ/4Tnwn/wBDRov/AIMIv/iqP+E58J/9DRov/gwi/wDiqzLv4reC7W4W3i1yK/uH4SHTke6Zj7eWGFRx/EZ7lC1p4L8VyjOFLaesQb3+dwce5Fdqy/F2u6bS89PzsTzx7mv/AMJz4T/6GjRf/BhF/wDFUf8ACc+E/wDoaNF/8GEX/wAVWLJ4o8cXChdP8ANCzsoWW+1WBVUHqzKhZuPQc/yqVJPiYVG+18KBu4Fxc/8AxFP6jNL3pxX/AG9F/k2HMjV/4Tnwn/0NGi/+DCL/AOKo/wCE58J/9DRov/gwi/8Aiqw5bL4n6hJ5Umq+HNGgznz7O2luZRx0xIVXk1N/wg+t3TA6r4+1yUDkrZpBagn/AICmcUfVKUfjrR+XM/yVvxDmfY1v+E58J/8AQ0aL/wCDCL/4qo3+IHg2I4l8W6Eh9G1KEf8As1czqHgvwjb39vp3iHVNc1u7vJMR2t1qU8zEHgsUQgBR6kYGKm8O/CfR9J1ybUJ9O09IwzLb2kUPmLt7M7yZZm4BwCADzyeaymsGlaM5SflFW+/mv+B0RoT5eep7qtdXvr6afjsdzYajZarYx3ul3kF7ay58ue3lWRHwSDhlJBwQR9RVisDwXGkXh+aOJFRE1PUFVVGAoF5NgAVv1ymIUUUUAFFFFABRRRQAUUUUAYXh/wD5Dnij/sKJ/wCkVtW7WF4f/wCQ54o/7Cif+kVtW7QAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRVe/1Cz0uykvNTu4LO1jxvnuJBGi5OBljwOSBTScnZbgWKK41vit4Ue6mttNurrVpICA/wDZljNcqCeg3opU59jUX/Cb+I9RlC+HfAepPGFO+bV5ksQD2wp3M35Cu5ZdivtQ5f8AFaP/AKVYnnidvTJp4raFpbiVIo1+87sFA/E1w02n/EbxCHgvtQ0vw1YyOQ39m757oJjoJGwoJPcKCKmt/hP4aM0c+ufb/EVxEMJNrV49yVGc4APy4Ppiq+rYemv31XXtFc343S+5sXM3si/q3xG8K6O0cU2sQXVzI4SO0sT9pmYkZHyR5I47nistfiDrmpR7vD3w+1yc8c6m0divXrliSRjnpXUaZ4a0PRZTLo+jafYSMu1ntbVImI9CVArTo9tg6fwU3L/E/wBI2/MLSe7OAksfifrljNFc6pofh1ZyQps7eS4nhTPTczBS2OMge4wek9n8JPDCXwvtaS88R3wG0XOtXJuSFznbtOFx7ba7iih5jiEuWlaC/uq347v5sORddStY6bY6ZAINNsreziUYEdvEsagemAKs0UV58pOTu2WFFFZ1/qy28d1Hp8Q1HULdFb7DFKqyHccAnJ+Ud8nsD1qW0lqXCEpu0f6/yLtxcQ2lu891KkMMa7nkkYKqj1JPSsG9vNV122tD4VubaCwukLvqbgsyDOMRxkck+p49qkg0WfWdPi/4TKCzupUuPPjt4Q3lRcYCnJ/eY5OSMZPTgVvABQABgDgAdqjWfkvxOm9PDvT3pJ+sf+D89PJlW3060truW7jgjF3OFE1wEAeXaMDJH0q1RRVpJbHLKUpO8mYXg/8A5Adx/wBhTUf/AEtmrdrC8H/8gO4/7Cmo/wDpbNW7TJCiiigAooooAKKKKACiiigDC8P/APIc8Uf9hRP/AEitq3awvD//ACHPFH/YUT/0itq3aACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooqnqer6dotn9r1i/trC33bfNuZVjXPpknrx0qoxlJ8sVdgXKK4mb4seHJLj7NoIvvEVz5fmGLR7Vp9g6fM3Cjn1P8xVTTtJ+IXiGzkvta8Sf8I0Lkt5el2VnDK1unQBpWyS2OeMYJ7dB3rL6sY81dqmv717v0STfztbzI5101PQawvEvjPQvCMcB1y98mW6bZb28cbSSzN6Kigk9hnpkjnmsNfhTpdw7nXdc8Ra4j/8sr7VXEa/RY9mK2dC8CeGfDV615oukw290ybGuCzSSFfTcxJ7ChQwMHeU5S8kkk/m22v/AAEPeZjxeOtf1aEN4e8B6sdxIEmryR2Sj3wSzH/vnmnFPiXqMxUzeHdEgyMNGst5L79di4rtqKn63Tj/AA6UV63l+bt+Acr6s4f/AIQXxBqDFvEPj7V5RuJWPSo0sVUHoMruY492p9v8I/BqXK3V9psmq3QYsZ9SupLhnJOeQzFT19K7Wih5li7WjPlX920f/SbByRI4LeG1iEVtDHDGOiRqFA/AVJRRXC227ssKKKKQBRRRQAUUUUAFNeRIwDIyqCQo3HGSegrI1zX5NNmistO0641HUbhS0UMY2ooHBZ5Dwqj8T7Uknhq2vtcj1bVXlupIdrW1rI+YbZgOWVcDLZ53HJ9MVDlraJ1RoJRU6rsmnbq38r7eb00drvQZbajquratPAmmvZaVGHie5uGKTSv0zEo6KD/Eevarmi6Dp+gWrQ6bBsMh3SzOS0kzf3nY8sfrWjRQo2d3qyZ1248kFyx00V9bdX3f9IKKKKs5wooooAwvB/8AyA7j/sKaj/6WzVu1heD/APkB3H/YU1H/ANLZq3aACiiigAooooAK5Lxz9g87Qv8AhIfs/wDYf21/tn2vHkZ8iTy/M3fLt3Yxnjds74rraKAPMLyeOb4V6LbXt1DGq3VuJXvx5kUMBYtEblCQShQKMMVySuSOa6X4ehh4FijgaPCz3Swyop8l18+Ta8a54iIwUXPC4AYgBj1VFAHNeEUvI9Q8TLqU8FxcjVV3yQQmJD/odtjClmI4x/Ef6V0tYXh//kOeKP8AsKJ/6RW1btABRRRQAUUUUAFFFFABRRRQAUUUjMFUsxAUDJJPSgBaK4vVviz4S0y8axtr6TV9SztSx0qFriWRs/dBX5c+xYdKz7HSPiB4pge81vxC/he2uCwTSrC1jaaKPcdpadskOV/u+3Q5FejHL6qh7Su/Zx6c19fRJNv1tbzI51stTr9a8UaF4dC/27q9lYF1LIlxOqs4HUhTyfwrlB8UZNabb4C8L6n4iXJxdsBaWhwcMBLIOSPTFbPh/wCHfhvw47T2tj9qvpFAmv75jPPKQMZLN0J77cD24rp6fPgqXwwc33k7L7lr/wCTIPefkcJ9g+JGuuv23VtL8M227Jj0+E3c7DP3S8gCrx3AP9Km0r4UeG7K7W+1ZLrxFqIGPtmtzG6cDsAG+UAduMj1rtaKl5hXUXGlaCf8qt973fzbDkXUbHFHEgWJFRR0CjAp1FFeeWFFFFABRRRQAUUUUAFFFFABRRRQAUUVjjxJZ3WsS6Rpjm5vI0YyPHGXht2xwJGHAJPbOfpScktzSnSnUvyq9tX5LzL+o6jaaTp8t9qM629tCMvI/QdvxOeMVjF73xdowNtJf6DbvNgs0arPPDjqvOY8k9SM4HTmrOiaNeWZmuta1KTUby52+YuNsEQByFjj7Yz1OSa2aiznvoux0c9Og7U/ekn8XT5Jrv1f3IrafYW+l6fDZWSFIIV2oGYsfxJ5NWaKKtJJWRyyk5Nyk7thRRRTJCiiigAooooAwvB//IDuP+wpqP8A6WzVu1heD/8AkB3H/YU1H/0tmrdoAKKKKACiisrxLNfW+gTSaWzpcb4gXjQO6IZFEjIpBDOELFRg5IAwehANWiuT8DatrGrWqyavHdxiPTrNZBeWv2eRrra/nnYVU7c7MHG3IbbxXWUAFFFFAGF4f/5Dnij/ALCif+kVtW7WF4f/AOQ54o/7Cif+kVtW7QAUUUUAFFFFABRRWfrOvaV4esTea5qFvYW4zh55Au4gE4UdWOAeBkmqhCU5KMVdsNjQorz26+ImqeIG+y/DXQptTZlXdqmoI1taQbjwcOA8mByQo6YIz0pYvhadZuzefETXbrxK4/1VmAba0h6HiJD8zA5G4nkHkEjNeh9RVJXxU+TyteX/AIDpb/t5ryI5r/CXvEXxCWx1uDQfCum/8JJrcytI9rb3SRpbRgD5pZDkJncMAjn2yM50Pw91HxWkl18TNSmuRO2RolhcvFZQqMbQcYZ2GM5Jxk8dAa7HRPDei+G7d4dB0u1sEkxv8iMKXx03Hq341p1X12FBcuDXK/5n8T9NbR+WvdsOW/xFPS9I07RLFbPR7G3sbZTkRW8QRc+uB1PHWrlFFebKUpPmk7ssKKKKkAooooAKKKKACiiigAooooAKKKKACiiigAqjq+tafoVl9q1S5SCPO1QeWduyqo5Y+wqje6zc3tk3/CIiz1GcXBt5JXnHlW7AZLNjlsccDnmrFpose+zvNWEN/qttEYxemEKRk5O0fw1m5N6R/wCAdcaMadpV9uy+L/gL116pMrz22uajrkbrepYaRCUkVYVzPdHAJD7hhFzxgcn1Fa9va29ojJawRwqzl2EaBQWJyScdye9S0VSik7mVStKaUdku36936/LQKKKKoxCiiigAooooAKKKKACiiigDC8H/APIDuP8AsKaj/wCls1btYXg//kB3H/YU1H/0tmrdoAKKKKACuZ8e2Md54ejaWe/jMV5bGKOymWIyymeMRhiQQBv289uTXTVyfjLV76wvtOtbaSS2tp0mlluU0mW/w8bR7Iysf3c72OT/AHOMdaAM74bX/wDaNxfXNxbzx3U9lZziW41L7W7wv5pjU4RQm0+Zx1yT7Z72uK+G0n2qxur+dJLa/vRHcXlmNJayit5GBJVSyK0rdQzlmzgfdBAra8QalfWtzpmn6QbeO71GdoxNcxNIkSJGzs2xWUsflAA3D72e2CAbdFcfdeNJYPAun6sYoor2+nS0ClHljjmLlHIVfmcDaxCjluBxnNbmh6n9v0GO8ku4rxhvEklvayQ8qxBXymLOrDGCpJOQfpQBW8P/APIc8Uf9hRP/AEitq3a5rwjexahqHia5gWdI31VcCeB4X4s7Yco4DDp3FdLQAUUUUAFFFeda74ivvGPipfCXgrUHtre22zazrFoyt5CHO2GNuf3jEHntjvgiurDYaWIk0nZJXbeyXn+i6vREylYu+LviPHpF9NoHhiwn17xN5YKWVtGWS33EBXmfoq/MCee4yVB3VB4W+FWmWX/E38XxxeIPEt03nXd7dr5iK+B8kaEbQq4wDjP0GFHSeFvCGj+DtLNlolt5Yc7pp5Dulnb+87dSeT7DPAFbddc8ZGjB0cHdJ7y2lL7tl5fe3ZC5bu8goooryiwooooAKKKKACiiigAooooAKKKKACiiigAooooAKKCQqkscAckntWJc6zfvryaZpWlSTLGym7vJ8xwxoecIcZdsdhwO5qZSUdzWnSlVbUemvb8zUuLuG3YIzgzMrNHCGG+TaMkKCeaxLWDU/Etjdx+I7JbDTrlVWKzSVhcbQckyOpAGePlHbIJq5p/hrT9P1S41PEl1fzsSbq5fe6KTwi/3VHTA/HNa1TyuXxfcbe0hR0o6vT3tmmu2v47+hDZ2Vtp9nHa2MEdvBEMJHGoVVH0FTUUVpscrk5O73CiiigQUUUUAFFFFABRRRQAUUUUAFFFFAGF4P/5Adx/2FNR/9LZq3awvB/8AyA7j/sKaj/6WzVu0AFFFFABWdr2qto2jyXiQrM4kiiRHk8td0kixgs2DtUFgScHAB4NaNZfiRQ/h66QteKHUL/oUCzSnLAYCMrKQehyMYJzgcgApeFvFQ8SImLdImOm2d65jm3hGnVyYzwMFdgPuGBwKv6zosesJbH7VcWVzaTedb3Vts3xsVZDw6spBVmBBB656gGsfwLZ6vp9ndW+paVY6bZh1a0S3gjglfOdxljjZowfu4KtzzkDHPVUAYX/CJ2i6FDpUV1dRQ2xie2kUoZIHjOQ4JU5Ynk7twOTxg4q9o+kQ6LYG2hlmnLyvNLNOQXlkdizMcADknoAAOgAq/RQBheH/APkOeKP+won/AKRW1btYXh//AJDnij/sKJ/6RW1btABRRSO6xozuwVVGSxOAB60Acb8S9fvtN0O20fw+0Y1zXp/sNlvk2+UGHzzcc4Re4zglTg9DteFvC2meEdBt9M0m2hiEcSLLMkSo1w6qAXcjqx681yXge1/4THxZd/EO9KSWzB7HQ4iAfJgRmV5enDOwb3AJGSDx6NXq4x/V6awcd1rP/F2/7d29bkR1fMFFFFeUWFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFNd0ijaSRlRFBZmY4AA7k0AOrN1vXLbQrRJrlJ5pJX8uCC3jLyTPgnaoHfAPXA4qhBrreJrS9i8MyyQBAFi1SW33Qu2fm2Akb8Adfu59a0dF0dNGsjAtzc3ckjmWWe6lLvI5ABPoBwOBgCs+Zy+H7zs9jGi/3+6fw7P5vp+fktzNutEk8U29nJrv2mzt9m6fSUlBR2zx5jrywA/hziuhRVjRURQqqMAAcAUtFUopa9TGpWnUSi9Iq9l0V/wCt3r5hRRRVGIUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAYXg//AJAdx/2FNR/9LZq3awvB/wDyA7j/ALCmo/8ApbNW7QAUUUUAFcf8Rpo4dFjLWmi3ILx+eNXukhRbfz4fM+8jAgnYD6EqRk4B7CsjxRc3Nl4enu7O4ht3tpIpneeURIY0kVpFLnhdyBlyfXt1oA574ePpk13qk2jQeG7WEpArW+hXS3BDAyHfIyonUEADB+6xzzXcVx/gS8ur9724m1qLVIpI4nkWKdZBbXJMhkjUAAqgUxBc9cE9znsKACiiigDC8P8A/Ic8Uf8AYUT/ANIrat2sLw//AMhzxR/2FE/9Irat2gArg/i1dG48M2/he0kkTUPEl1HYw+Um8pHvUzOR/dVM5+v4jvK880j/AIqz4zapq7Evp/heL+zbLGSjXUgzO4PZlGIyOexr0suSjVeIltTXN89or/wJq/lcie1u53tnZ22n2UNnYwR29tAgSKKNdqoo6ACpqKK85tt3ZYUUUUgCiiigAooooAKKKKACiiigAooooAKKiubmCztpLi7mjghjG55JGCqo9STWFeXeq6/bWh8LXVvb6fdIXfUmUtIozjEcZGMn+8ePaplJL1N6VCVTXaPd7L/g+S1L9/4h07T9StdOmmL3t0wWO3hQyPgn7xA+6o7seKr2el6tLrMt9rOpBoFLpb2FsMQhDwGkJ5diMew7VpW+n2ttcy3UUEYupwomnCAPLgYG4gc8CrNLlbd5F+2hTjy0lq1Zt2f3aafn6XsIiLHGqRqERRhVUYAHoBS0UVZyhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAGF4P/wCQHcf9hTUf/S2at2sLwf8A8gO4/wCwpqP/AKWzVu0AFFFFABXJ+OruaW3t9EsbYS3l2DdxO9ykCxfZ5InDZZWDHe0eFIweckCusrH8SW2hNp4vPEWm21/HbsFiWa1WdtzsqhUBBOWYqMDqcUAUvB0S3lr/AMJBJqbahc6paW7FjEkXlxAM8a7Fzg/vXyST19AK3L/UrHSrNrrVLy3srZSA01zKsaAnp8zECsjwvL4cdSfDenW9ibmzt711htVhLxSh/KLbRyflfjtz61H4qDwaloOpPbXFzaWN47zpbQPM6boJEV9iAs2C2OASN2egJoA6FZongEySI0TLvEgYFSuM5z6Y70y0vLbULOK7sLiK5tplDxTQuHR1PQhhwR7iuKtNL1mf4d6fo1naRLIrL9stL2SS3X7MxZvIEixuM42IwAOF3DIOK1fANtqEHw+0221O2/s+6SEoY0Ys0fJwfmQYPsRj60AWfD//ACHPFH/YUT/0itq3a5rwjby2moeJoZ7ye9kXVVzPOqB2/wBDtjyEVV46cAdK6WgDH8WeIYfCnhPUdbuVV1s4S6xs+3zH6Kme2WIHfrWZ8NfDtx4a8B2NpqKn+0p911fuxDO88h3NubuwyFz/ALNY3ii1l8Z/E/S/DUsEMmi6Kker37Oc+ZMS6QxEenBYg5BGR259Er061qGEjRXxT95+m0V+LfzXYhayuFFFFeYWFFFFABRRRQAUUUUAFFFFABRRTXkSIAyOqBmCjccZJ4A+tAbjqzb/AFdYI7qLTY11HUbdFY2MUyq/zHAzk/KO+T2FVLa/1bVtVnhGnNY6TGHia4nYpPM3TMaj7qj+8evGKuaNoWn6BaGDTIBHvO6WRjuklb+87Hlj9az5nL4djr9nCj/F1lpZLb5tPT0Wuu6KVvok+r6fF/wmUVneTx3H2iOCFG8qHjAU5Pz45OSOp6cCt4DAwOBRRVRikY1K06j126LovRBRRRVGQUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAYXg/8A5Adx/wBhTUf/AEtmrdrC8H/8gO4/7Cmo/wDpbNW7QAUUUUAFZPieIT+HblClsxyjL9qumtkVg6kN5qglCCAQQPvAdOta1cT8QtU1CNrTR9NsPtrXkElyE8hpRugntiA4H8BDuCMZPHIoAh+G8cVjdanpaxWhezgtVFxbaq99uixIqRlmRdgTYcKP72e+T3lct4JvtQuobuLUNCh0aOIoY0hhaMSE53HBA6YHT1rqaACiiigDC8P/APIc8Uf9hRP/AEitq2p54rW3knuZFihiQvJI5wqKBkknsAKxfD//ACHPFH/YUT/0itqyvitq0mmfD29t7NRJf6sV0yzhP/LSSY7MD3Cljz6V0Yai8RWhSX2mkKTsrlT4VRjUrHWPF8sEsc3iK/eeMzk7/syEpCpHQYAbp69cYrvaqaTp8ekaLZabAcx2dvHAhxjIRQo4/CrdVjKyr15VFt09Fol8lZCirKwUUUVylBRRRQAUUUUAFFFFABRVbUdRtNJ0+W+1GdYLaEZeRu3OB06nJxisYm98XaMpifUNBt3m5JVVnnhx26mPJP1wPeolJJ2W50U6EprnlpG9r/1u/T5lnXNem06aKy0zTZ9R1C4UtHEg2xoAcFnkPCgfifakk8M2t5rqatqjS3c0W029vK+6G1YDlkXAy2edxyfTFaOn2Ftpenw2VjH5cEK7UUsWwPqeTVmly3+Iv26pq1DTdN9Wn+WnRed20FFFFaHIFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAYXg//kB3H/YU1H/0tmrdrC8H/wDIDuP+wpqP/pbNW7QAUUUUAFRtbwtcpcNEhnjRo0kKjcqsQWAPYEquR/sj0qSigAooooAKKKKAMLw//wAhzxR/2FE/9IrauX2xeLvjnuz59j4Rs8AcbVvZj/49iNf+AsB0PXptCdY9Y8VPIQqrqaFiew+xW1Yvwihkk8DtrNyFW516+uNTmVM7VMj4AGT02qv+ea9LCP2VCrX62UV6yvf/AMlTXzIlq0juaKKK80sKKKKACiiigAooqhq+t6foVn9p1S5WFCdqL1aRuyqo5Y+wpNpK7LhCVSShBXb6Iv1jL4ks7zWJdJ0p2ubuJGMkscZeG3YDhZGHGSf4Qc8dqZPaa3qOuRyC+Sx0iApIkcC5muTgEhyw+Rc8YHJ9RWxBbQWqMltDHCrMXYRqFBYnJPHcnvU3lJ6aG/LSpRvL3m106eumvovm+hlaJot3ZNNdaxqcuo3tzt8zPywxgchY4+gAz1OSfWtmiiqjFRVkY1asqsueX+X4IKKKKZmFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAGF4P/wCQHcf9hTUf/S2at2sLwf8A8gO4/wCwpqP/AKWzVu0AFFFFABRRRQAVgeJLq9+36Npmn3slgdQuXSS6iRGdESJ3wodWXJKjkg8bu/I36o6ro9lrVvHDqCSMIpBLE8MzwyRvgjcroQynDEcHoSOhoAxbDUrjU/AFvqOoau2lPGjNdX0KRjiMsrMBIrKoO3PQ4Bq/4Zu9RuPCtvdatHK92wdgGjEckke9vLLLwFdk2EjgBiRxTbvwhot7o8elTW8y2MRiMcEV3LEqGL7m3awxggHjqQCckVqWVnFp9mltA0zxx5wZ53mc5OeXclj17n2oA838Ray1r8PfiRf3drPp7vOYRFOyF0aSyto15jZl5LDGD35xXf8Ah/TV0bw3pumIMLZ2kUA/4CgX+led+MnWRr7TXG5dT8ZadbMvqv2e1c59sIa9Ur0Knu4KnHvKT+5RS/UhfEwooorzywooooAKKguL2G3YRs6tOyM8cAYeZLtGTtBPNYlrban4ksbuPxNZR2Wn3QURWSSt5wUHJMjqQBnj5R24J61DlZ2W50U6PNHnm7R09fkuv5eZNea1c6hYt/wh/wBj1CcXBt5Jnm/dW5AyWbHLY4GB3P1qzZ6LGrWl3qvlX+qW8Rj+2tCEbk5O0D7v4VdtLS3sLSO1soI7eCIbUjjUKqj2AqahR6yKnWSXJSVl36v1f6LTvfcKKKKs5QooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigDC8H/8AIDuP+wpqP/pbNW7WF4P/AOQHcf8AYU1H/wBLZq3aACiiigAooooAKKKKACiiigDy2/Im+MGm2j8odfnnK54zHpUGD+BYV6lXl0ZE/wAfUiVubaa8nZc9msrFAf516jXfiv4VBf3X/wClyJjuwoorN1vXLfQrRJZ4p55Jn8uC3t4i8kz4J2gD2BOTgcV57aSuzWnTlUkoQV2zRZgqlmIAAyST0rFudY1B9fTTdK0qSSOJlN3e3BMcKKcHCHGXbHpwO5qG60J/FFvZy6/9otYAm6bSUmBjd88eYygFgB/DnFdAqqiBEUKqjAAGABUe9LyR0WpUd/elrp0XzT1/LbV7GVp3hvT9O1K41JVkuL+4Ylrq5fzJFUnhFJ+6o6YH45rWooq1FRVkYVKs6suabuwooopmYUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUVHcOyQ5Q4JZVBx0yQP60ASUVD5Mn/AD8y/kn/AMTR5Mn/AD8y/kn/AMTQBNRUPkyf8/Mv5J/8TR5Mn/PzL+Sf/E0ATUVnRX9lNbLcQ6zHJA03kLKksRUybtmwHGN275cdc8daW4v7K0iuZbrWI4I7TH2h5JY1EOQCN5I+XII6+tAGhRVGwurbVLRbrTNVW8t2JCzW8kciEjryARVjyZP+fmX8k/8AiaAJqKzp7+ytdQgsLnWYoby5BMFtJLEskoHXapGT+FGo6hY6PCs2r61FYRO21XupYolY9cAsBzQBo0Vmy6jYQX1vZT61DHd3Q3W8DzRCSYeqqRlvwq2u+O5WNpGkVkLfMBkYI9APWgDJ8H/8gO4/7Cmo/wDpbNW7WF4P/wCQHcf9hTUf/S2at2gAooooAKKKKACiiqOq6xZaLbxzag8iiWQRRpFC80kj4J2qiAsxwpPA6AnoKAL1FZs3iDTINDh1drrfZThDDJFG0hl3kBAqqCzE5GAATVnTtRtdVsI72wl82CTO1ipUggkEEEAggggggEEEGgDznw1bvP8AtGeMrrK+XZ2dtFjvulihb+UVemySJFG0krqiICzMxwFA6kmvPdC1LTtK+KnjFZmL32pajZwRwwxl5NqWcJ3tjog8w8njrXU2WlarJrMt/rOp74lLpb2FsNsKoeAXzy7EY68DnFb4qsqkoqOtlFemmv4m8MO1HnqPlT1V76+n+exDBrj+J7O9j8NyS2yKFWHVJLfdE5z83lgkFsAdfu59a0dF0eLRbEwR3FzdPI5klnuZS7yOQASew6DgYFX0RY0VEUKqjAUDAA9KWuVR6vcqpXunTpq0e27+btr6bdkFFFFWcwUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABUN1/qV/66J/6GKmqOeMyRYUgMCGGfUEH+lAHNX3gf7dfz3X/CUeJLfznL+Tb6hsjTPZRt4HtUH/AArz/qb/ABV/4M//ALGuq3XP/PKL/v6f/iaN1z/zyi/7+n/4mu1Y7EJWUvwX+RPKjnLHwP8AYb+C6/4SjxJceS4fybjUN8b47MNvI9q1vEd3e2Phu+n0m3e4vxCVtY0QtmVvlTIGflDEEnsAT2q7uuf+eUX/AH9P/wATRuuf+eUX/f0//E1hVrVKzvN3GklseXWOg6z4esZdLm0jFrbXWnalELCWS6DCN0Sf5jGhL4jEm0AlizEZJrQngubnXm8R/wBm30mlx6zFO0DWkgmeNLMxCUQkbztlYHG3Py7gDgZ9B3XP/PKL/v6f/iaN1z/zyi/7+n/4msRlO1u11LS7mfTILi1eTcENxbtbOz7cBtrpkdhllPToR1yNBsPFtvqqya5fedabSGT7bFLzjj5Vs4j/AOPj6Guj3XP/ADyi/wC/p/8AiaN1z/zyi/7+n/4mgDhfENldv4g1O2Sxupp9SuNNks7mO3Zo41hlDNukA2pswz4YjO/5ckkVP43jkvdR02eE6taQxQ3kL3unWDTzI52KI/LMbZR8Md+3+BcMN2a7Pdc/88ov+/p/+Jo3XP8Azyi/7+n/AOJoA8yj0zUobOPT7jR54b69h0cwGCF5IbXyGQyIZeQnllXYbjk7uNxJr01v+P6L/rm/81o3XP8Azyi/7+n/AOJoRJWnEkoRdqlQFYnOSPYelAGR4P8A+QHcf9hTUf8A0tmrdrC8H/8AIDuP+wpqP/pbNW7QAUUUUAFFFFABWB4ktb37fo2p6fZSX50+5d5bWJ0V3R4nTKl2VcgsOCRxu78HfooA4eLRdWtPCuixjT3mvNKvEvZLeOWPE2/fvjjYsBlPNON20EqOec1ueF9NubTQZo9Ria3mu7u6unhWT5oRNM7hdyn7wDDJU9c4PetyuK8aTSSasLaXUrrTraDR7y+SS2uGhzLGYwGJUjcEDZ2nKndyDgUAP0v4e6fa6xrV5M2pK15dI0Uses3Yd41giXDsJMnDq+M54x2xWr/wh+m/8/Otf+Dy9/8AjtZWpT3Gqy+GLPU7m5sIb62lnuvs1w9uxlWJSE3qQwA3O2M87OeAa2fB97c6l4J0W9v3MlzcWMMkrkY3sUBLY7Z6/jQNtvcj/wCEP03/AJ+da/8AB5e//HaP+EP03/n51r/weXv/AMdrdooEYX/CH6b/AM/Otf8Ag8vf/jtH/CH6b/z861/4PL3/AOO1u0UAYX/CH6b/AM/Otf8Ag8vf/jtH/CH6b/z861/4PL3/AOO1u1i+Mr2507wTrF5ZSGKeG0kdJQMmL5fv/wDARz+FAGffeDI2vNNNjeawsC3JN4Drl580XkyAAZl/56GM8en1q7/wh+m/8/Otf+Dy9/8AjtU/DqGHU/EGkx3t1PY2xh8qWe6eaSJniBdRI5LcfK4yeN/HGBSeDA0txqV3Y3F5NocpjWwa8uZJ2lZd3mSo0hLeW2UC84OwsOGBIBd/4Q/Tf+fnWv8AweXv/wAdo/4Q/Tf+fnWv/B5e/wDx2t2igDC/4Q/Tf+fnWv8AweXv/wAdo/4Q/Tf+fnWv/B5e/wDx2t2igDC/4Q/Tf+fnWv8AweXv/wAdo/4Q/Tf+fnWv/B5e/wDx2t2igDC/4Q/Tf+fnWv8AweXv/wAdqlo3gyOPQbBNYvNYk1BbaMXTrrl5hpdo3niXH3s9KyZb27/4SqW8F9dC5j8QR6clmJ28o25t1Yr5WdpOGaXdjdx1wMU3xV4hu4fHuj2udWtLS2v4oiILO4MV4ZI2JJdVKsq/KNueu8kfIDQB0v8Awh+m/wDPzrX/AIPL3/47R/wh+m/8/Otf+Dy9/wDjtbtFAGF/wh+m/wDPzrX/AIPL3/47R/wh+m/8/Otf+Dy9/wDjtbtFAGF/wh+m/wDPzrX/AIPL3/47R/wh+m/8/Otf+Dy9/wDjtbtFAGF/wh+m/wDPzrX/AIPL3/47VKLwZGNeunlvNYOnm2hECf25eZEoaXzD/rc8qYvy+tTeKi8+paDpr3NxbWl9eOk7207wu+2CR1QOhDLkrnggnbjoSKwZ9X1ab4Z6I8Ess899dw2j3InMLSRGQqrmQDK7wFBZRn5yV5xQB0v/AAh+m/8APzrX/g8vf/jtH/CH6b/z861/4PL3/wCO07wldpd6AFSKaJra4mtpEmu3uiHjkZWxK/zOMg4J5xxgYwNugDC/4Q/Tf+fnWv8AweXv/wAdo/4Q/Tf+fnWv/B5e/wDx2t2igDC/4Q/Tf+fnWv8AweXv/wAdo/4Q/Tf+fnWv/B5e/wDx2t2igDC/4Q/Tf+fnWv8AweXv/wAdqlq3gyN7OMaXeawk/wBpgLE65ef6oTIZRzL3j3j8eK6W6kkis5pIE8yVI2ZE/vEDgVxPga5nN/ZKNRur+O/0OC/uWuLhpdszN95ck7A+W+VcKPL4A5oA3v8AhD9N/wCfnWv/AAeXv/x2j/hD9N/5+da/8Hl7/wDHa5TwRfalJ4otJb28mmttTsrueFjeyTC52TxgO0TfLblVbAVMg7juIKgH0igDC/4Q/Tf+fnWv/B5e/wDx2j/hD9N/5+da/wDB5e//AB2t2igDC/4Q/Tf+fnWv/B5e/wDx2j/hD9N/5+da/wDB5e//AB2t2igDC/4Q/Tf+fnWv/B5e/wDx2j/hD9N/5+da/wDB5e//AB2t2uZeN7f4pW2y6uzFd6VcySW73DtCGSS3VSsZO1Thm5AydxoAveFtJk0TQzYy7jtvLuVC8rSsUkuJJEJZiSTtYZJJOa2K8/129u18RajdLfXUU+n3+m29papOyxyRTSIH3Rg7X3b5FywONmRjBr0CgAooooAKKKKACiiigAqpqGk6dqyxLqmn2t6sL+ZELmFZNjf3huBwfcVbooAo3miaVqFq9tf6ZZ3UDy+c8U1ujq0n98gjBb361dVQihVAVQMAAdKWigAooooAKKKKACkdFkRkdQysMFSMgj0paKAM+38P6NZ2gtbTSLGC3XfiGK2RUG8YfgDHI4PrS6XoOkaGsg0XSrLThLjzBaWyRb8ZxnaBnGTj6mr9FABRRRQAUUUUAFZ+p69o+iNEus6rY6eZgxiF3cpF5m3G7buIzjcM46ZHrWhXLa5pd7f+P9EmtLu9sIodNvlkubWKNhlpbQrGxkR1G4KxHAJ2HB4NAGveafocc0utahaaeskduwlv54kBWHBLbpCOExnOTjGauNZ2zpAj28TLbsGhUoCIiBgFfQgEjj1ryTU/+Ekv4/E8As9UMN9oeqo9m1vdOqTghYVR5HZHZlL4ESqpBxzgY2dWbxDFruvLoY1K8upradraV1uYo7Rgi7UVXIt5ckfKy4bJ57mgD0iqGna9pGsSSx6TqtlfPD/rVtbhJSnbkKTjoevpWH4Fiu4re9+06je3kTSKY0vLG6tzEdvzAG5d3YHg9doPTvXK2XhzW7XwvpGp3k10rW0K2klrYWskNzFavOjy5IYuz4jX7gUhd+0ZNAHqtQxXdtM+yG4ikbLjajgnKnDD8CQD6E15pM2tQwWFwBrl3FHczi109xeRyTQmYeW0kyfdYKDgT8FSN2DuNWrPT7tvHWmajrEesMkc2qwRyI1yVXddwtAGCHAjKK/LfIQq56LQB6JBPFdW8c9tKk0MqB45I2DK6kZBBHBBHen15npCeKVtfDOn3P8AaxGrWFhLe3Mpk3WckCBrhZGPKGXEaYOCSZD1zVzwVFro8QPLrmo3xuPLmF1aSWFysJbzBtZZXkaHgdBEFyG5HHAB09/eeGtXvX8Panc6TfXWQX0y4kjkfOA4zEcnoQ3TpzVu40zSvsM8V1ZWf2WVFSZJIk2OqjChgRggdAD0rhbzTNZXxZqV+1sZtJj8SWty1ulq/nuPsdvF5yPkhkVvvKFzhH+bgrWFeQeItYg1q2ntdXFtd6bNNLa+XeDyZ454ysaSPId7FS/+qCqw4AbHAB6/aWltYWsdrY28VtbxDbHDCgREHoAOBU1eatFq8ni6A2+o6vZ6cstqdPV7C+n8yH5fMWVjIFUlt4JnQsFIIPTHpVABRRRQAUUUUAFULbQtIsmkaz0qyt2kmE7mK3Rd8g6OcDluTz15q/RQBStNG0vT7ye7sNNtLW5uTmeaGBUeU5z8zAZPPPNXaKKACiiigAooooAKzLjw1oV3qn9p3Wi6dPf4x9rktEaXGMY3kZ6cdeladFAFJ9F0uS+trx9Ns2urVNlvOYFLwr6I2MqOTwKu0UUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAf//Z)

*Figure 2: Python model ROC Curve*

A close up of a map

Description automatically generated

*Figure 3: SAS model ROC Curve*

A picture containing clock

Description automatically generated

*Figure 4: Python Correlation Matrix*

|  |  |  |  |
| --- | --- | --- | --- |
| Model | AUC ROC Score(%) | Mean Accuracy(%) | Prediction Accuracy(%) |
| Full model | 79.75 | 74.55 | 81 |
| Reduced model (13 var) | 79.75 | 74.60 | 81 |
| Reduced model (3 var) | 75.52 | 71.50 | 69 |
| Reduced model (1 var) | 59.92 | 59.92 | 89 |

*Table 1: Python model results*

|  |  |  |  |
| --- | --- | --- | --- |
| Model | AIC | AUC ROC score | Accuracy |
| Imputed Categorical | 15482 | 58.02% | 90.05% |
| Encoded Dummy | 12664 | 58.04% | 88.65% |

*Table 2: R model Results*