

Frank Enendu

HubPay Assignment

About assignment

This is a short report or slide deck showing my findings on retention versus churn (departure) of HubPay customer dataset

The objectives of the assignment was to:

- Descriptive analytics
- Data Prep and cleansing
- Build classification models to predict churn
- Evaluate the performance of the models a
- Build logistic regression model(s) explaining churn in terms of the explanatory variables

Created Data Visualisation using Power BI

Descriptive analytics

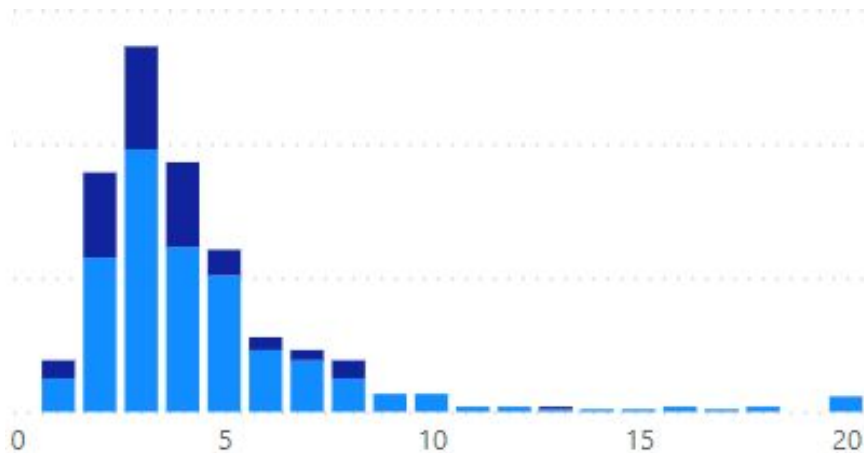
To perform descriptive analysis, I employed the use of Power BI to help me filter and disaggregate the data for reasonable insight.

Here is a quick link to the visualization dashboard.

[Link](#)

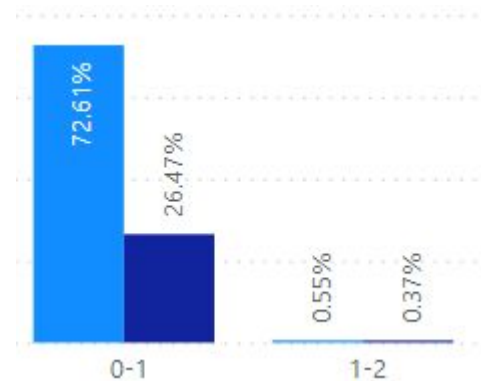


Findings



Churning verses Credit Score

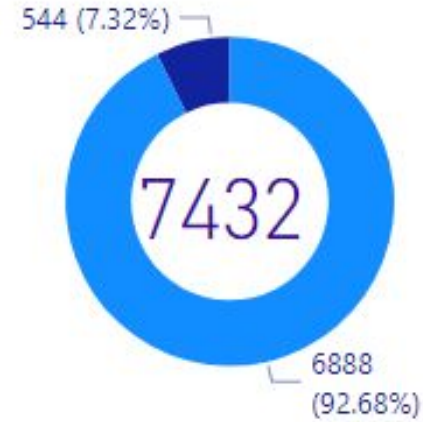
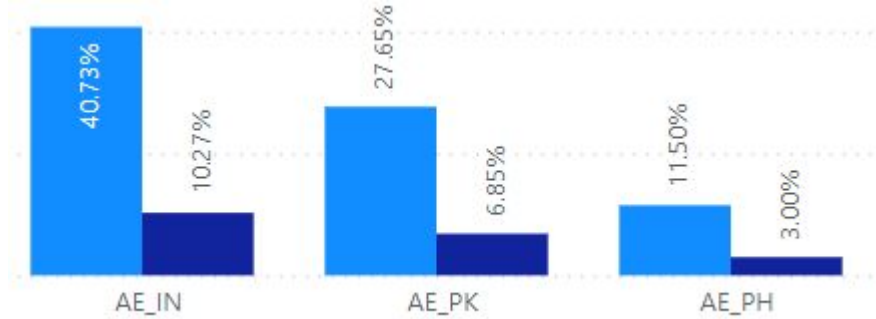
Customers who churned have low credit score. NB: The higher the credit score, the lower the risk of churning



Churning and Tenure

Most of the customers who churned are customers who are one year or less.

Findings



Country Pair Distribution

The Countries pair between which funds are mostly remitted are Arab Emirate and India.

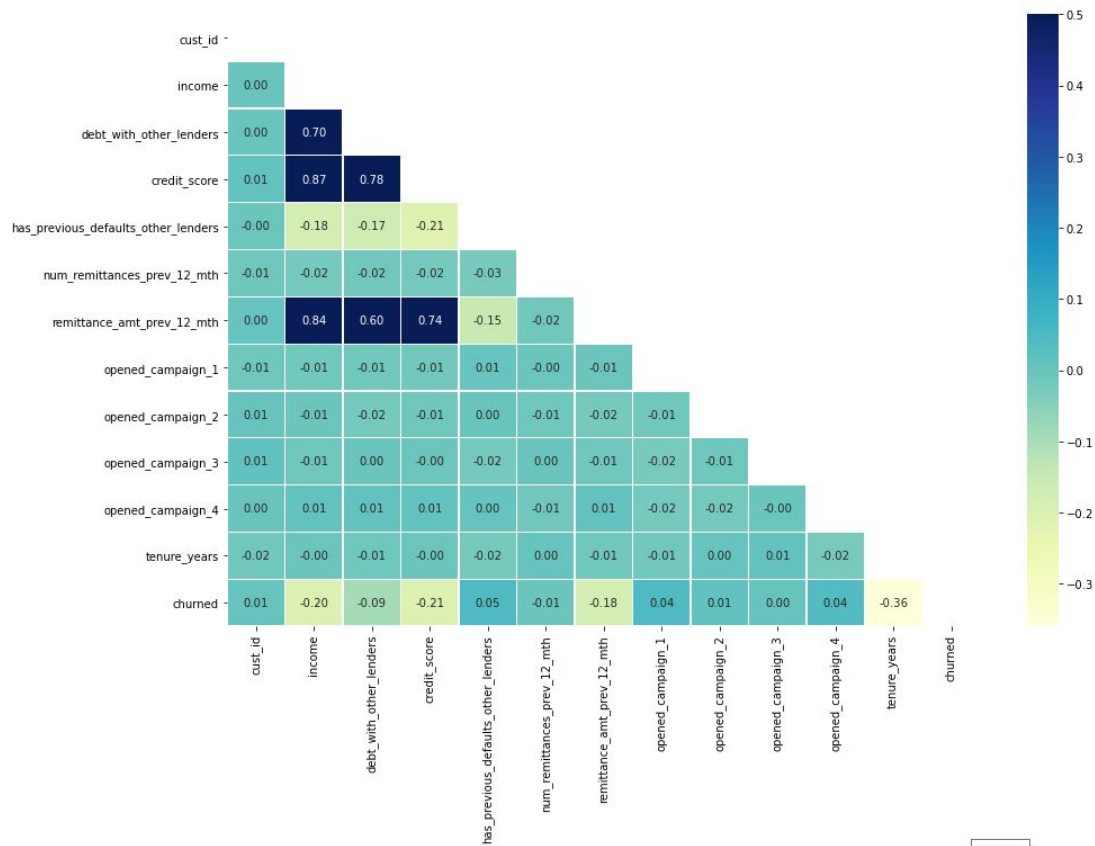
Customer Distribution

We happen to have majority of our customer still sticking around

Findings

Relationship between customer churn and other variables

From the result of the correlation heatmap, we see that churned is either negatively correlated or has very poor correlation with all other features. So when Income is increasing, it is less likely for customers to churn, when credit score or debt is high, there is less likelihood for customer to churn



**Prepare and cleansed my data using pandas,
numpy, sklearn, and other libraries**

Data Prep & Cleansing

To get data ready for modeling, I employed the following steps;

- Imported libraries
- Read the dataset
- I converted all dataset to numerical value, including hot encoding the categorical variable, (main_remittance_corridor).
- I dropped rows containing null value (a total of 794 rows)

```
22]: def print_unique_col_values(df):
      for column in df_no_na:
          print(f'{column}: {df_no_na[column].unique()}')
      print_unique_col_values(df_no_na)

cust_id: [ 1 2 3 ... 7430 7431 7432]
income: [63863.13588 51537.47964 3298.248451 ... 46424.99755 28140.26622
14095.82627 ]
debt_with_other_lenders: [87983.13439 63655.10915 4776.336091 ... 24527.67428 58965.30648
13166.6542 ]
credit_score: [20. 17. 2. 5. 3. 7. 1. 8. 9. 4. 6. 12. 19. 11. 13. 14. 16. 10.
15. 18.]
has_previous_defaults_other_lenders: [0 1]
num_remittances_prev_12_mth: [ 22 20 26 18 21 13 24 25 31 28 27 30 29 23 316 16 12 34
8 19 33 324 17 32 35 327 15 347 339 14 11 314 328 329 323 9
36 330 349 37 40 336 341 310 344 348 345 325 311 303 309 10 338 321
337 38 300 313 302 340 308 322 342 304 333 334 317 343 306 335 346 310
320 307 301 312 318 305 332 350 39 315 331 326 7]
remittance_amt_prev_12_mth: [23377.33823 8353.525522 1213.782465 ... 22261.95628 6162.548544
4289.214953]
main_remittance_corridor: ['AE_IN' 'AE_PK' 'AE_PH']
opened_campaign_1: [0 1]
opened_campaign_2: [0 1]
opened_campaign_3: [0 1]
opened_campaign_4: [0 1]
tenure_years: [2.06525811 2.76167614 0.2970638 ... 0.48482528 0.49382951 0.425786 ]
churned: [0 1]
```

Data Modeling

Declaring Dependent & Independent Variables

The independent variables includes;
Income, debt_with_other_lenders,
credit_score,
has_previous_defaults_other_lenders,
num_remittances_prev_12_mth,
remittance_amt_prev_12_mth,
Main_remittance_corridor,pened_campaign_
1, opened_campaign_2, opened_campaign_3,
opened_campaign_4,
Tenure_years.

While churned column became the
dependent variables.

```
In [35]: X.head()
Out[35]:
```

	income	debt_with_other_lenders	credit_score	has_previous_defaults_other_lenders	num_remittances_prev_12_mth	remittance_amt_prev_12_mth
0	63863.135880	87983.134390	20.0	0	22	23377.338230
1	51537.479640	63655.109150	17.0	0	20	8353.525522
2	3298.248451	4776.336091	2.0	0	26	1213.782465
3	14402.605700	13925.390670	5.0	0	18	6202.880445
4	8635.683507	10143.513660	3.0	0	21	6175.393029

```
In [36]: y = hot_encoded["churned"]
In [37]: y.head()
Out[37]:
```

0	0
1	0
2	1
3	0
4	0

```
Name: churned, dtype: int64
In [38]: # plt.scatter(X,y)
# plt.xlabel("Independent")
# plt.ylabel("Dependent")
# plt.show()
```

chrome

Splitting Dataset

Splitting Test and Train dataset

```
In [39]: X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2)
```

```
In [40]: X_test.shape
```

```
Out[40]: (1328, 14)
```

```
In [41]: y_train.shape, y_test.shape
```

```
Out[41]: ((5310,), (1328,))
```

Splitting Data

I splitting dataset to test and train dataset in the ratio of 8:2.

Data shape for **((5310,), (1328,))** test and train respectively

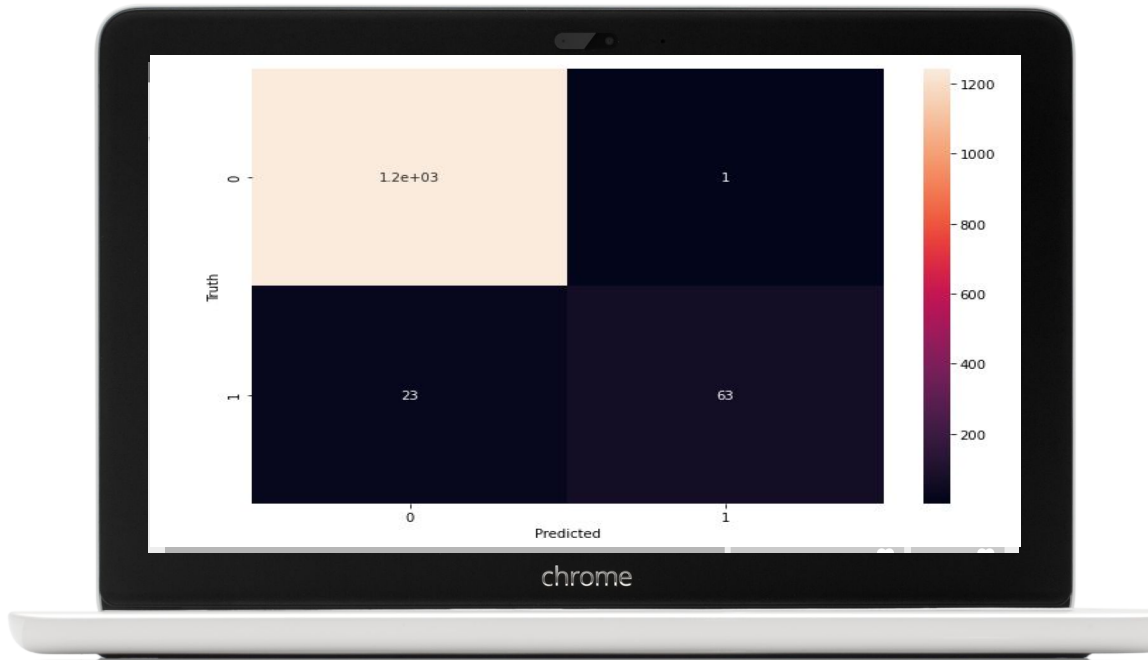
Building Prediction Models using RandomForest, XGBoost, and Logistics Regression

RandomForest

Our model has an accuracy of 97%, This simply means that if we make prediction 100 time, our model is likely to make the right prediction 97 time. this is pretty good. Other model evaluation metrics are shown below

```
print(classification_report(y_test,pred))
```

	precision	recall	f1-score	support
0	0.98	1.00	0.99	1242
1	0.98	0.73	0.84	86
accuracy			0.98	1328
macro avg	0.98	0.87	0.92	1328
weighted avg	0.98	0.98	0.98	1328

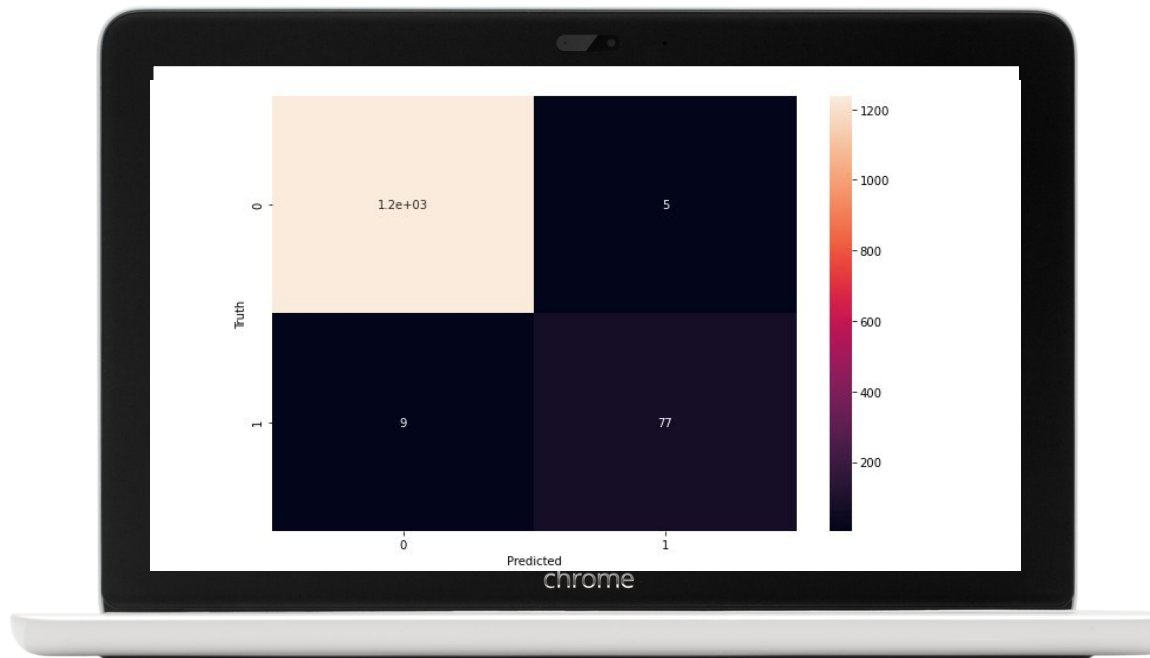


XGBoost

Our model has an accuracy of approximately 98%, This simply means that if we make prediction 100 time, our model is likely to make the right prediction 98 time. this is pretty good. Other model evaluation metrics are shown below

```
print(classification_report(y_test,xg_pred))
```

	precision	recall	f1-score	support
0	0.99	1.00	0.99	1242
1	0.94	0.90	0.92	86
accuracy			0.99	1328
macro avg	0.97	0.95	0.96	1328
weighted avg	0.99	0.99	0.99	1328

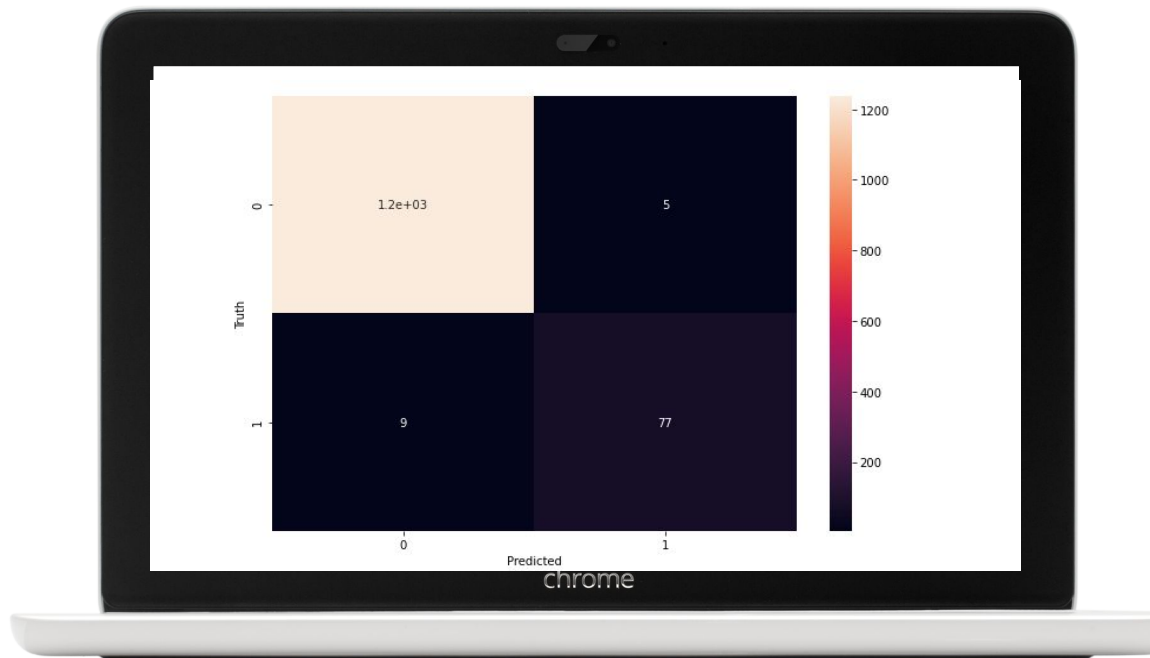


LogisticsRegression

Our model has an accuracy of approximately 94%, This simply means that if we make prediction 100 time, our model is likely to make the right prediction 94 time. Other model evaluation metrics are shown below

```
print(classification_report(y_test,log_pred))
```

	precision	recall	f1-score	support
0	0.98	1.00	0.99	1242
1	0.97	0.71	0.82	86
accuracy			0.98	1328
macro avg	0.97	0.85	0.90	1328
weighted avg	0.98	0.98	0.98	1328



Created and deployed model on a landing page

Landing Page

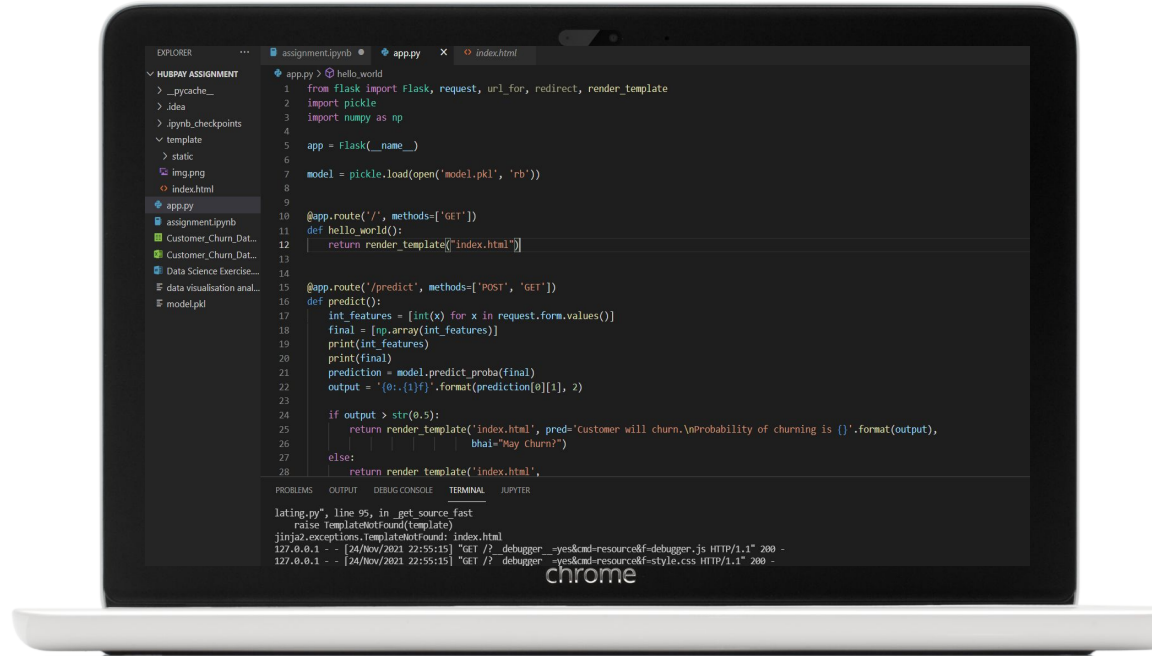
— — —

I created a landing page with a form where i can use to collect new data for potential prediction



App Deployment

I created a backend deployment route, where i connected and deployed my model.



Thanks

— — —