# Product recommendation based on purchase history

IE 7275 Fall-2020

Final Project

Tushar Sharma

# Problem definition

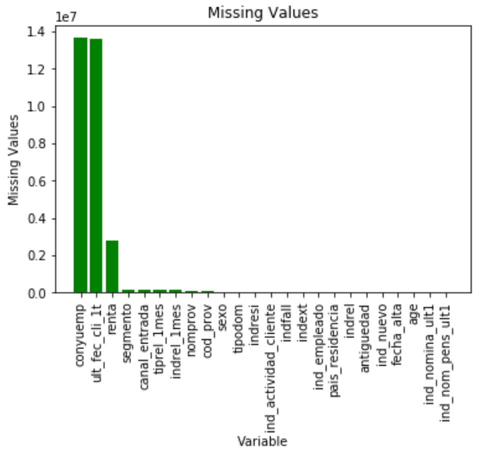
- We are given 16 months of past purchase history of financial products of a bank's customers
- Product offerings are such as credit card, guarantees, loans, pensions and investments
- Need to recommend them products they might be interested in
- Customer demographics like age, rent, employment, residence
- Find top 7 products to be recommended to each customer

# Data Description

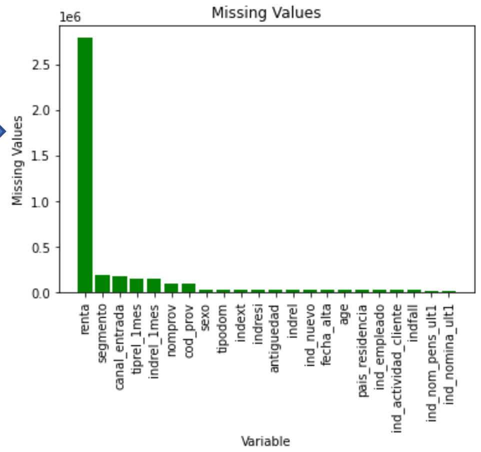
- The downloaded files from Kaggle are training and testing dataset with information of ~950K customers and their 16-month activity


- 20 demographic features and 24 product purchase binaries split by purchase month
- Almost all demographic features have missing values

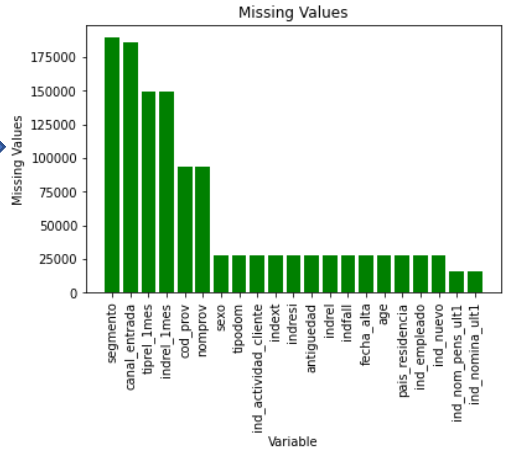
# Preprocessing – Missing values & Outliers




1st 2 cols - Fill NAs

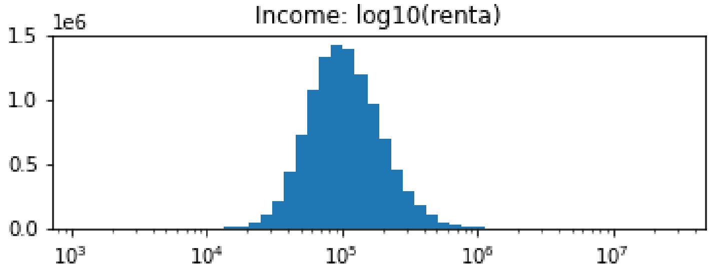


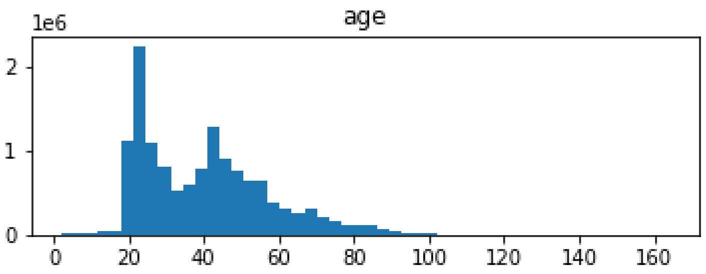

Interpolate income using simple linear regression




Interpolate variables with mode

D
O
N
E

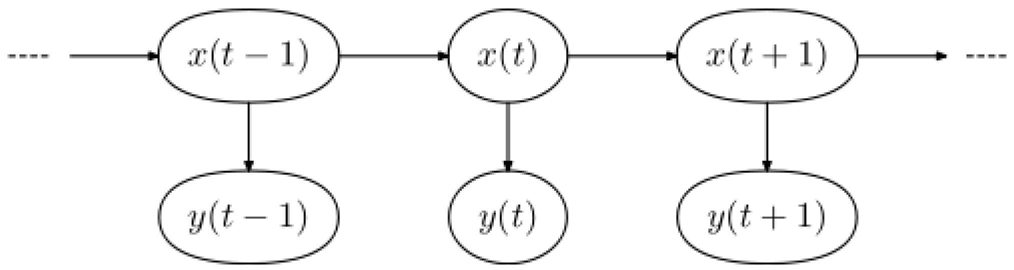


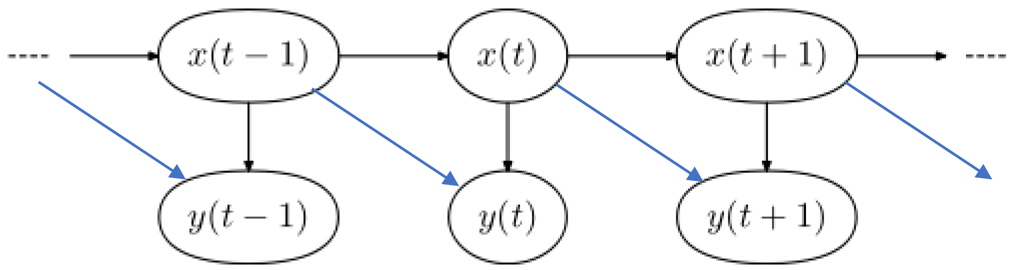
# Markov Assumption

- A Markov chain is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event.

Discrete time MC of order 1




Discrete time MC of order 2



# Model Formulation

- $X(t-n)$: Demographics & Products present in month ‘t-n’
- $Y(t-n)$: New products added in ‘t-n+1’ = Products present in month (t-n+1) – Products present in ‘t-n’
  - ‘n’ = {0,16}
  - State ‘t’ is the last month for which Y is given
  - State ‘t+1’ is the last month for which X is given
- We need to predict $Y(t+1)|X(t+1)$ if we assume markov-chain of order 1 and $Y(t+1)|X(t+1), X(t)$ for order 2

# Preprocessing – Data Preparation

- We need to create X and Y such that they follow our assumptions

Purchase Month

CustomerID

| Demographics | Product purchased |
|---|---|
| ……………………………… | ……………………………… |

$$X=\begin{bmatrix} . & \mathit{Demographics} & . & . & . & \mathit{Product\ purchased} & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \end{bmatrix}$$

$$Y=\begin{bmatrix} . & . & . & \mathit{Added\ product} & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \end{bmatrix}$$

- Take a month's data and stack in X, and products in next month and stack in Y
- Subtract Products in Y with corresponding products in X to get 'added products'

# Evaluation criteria

- Accuracy

$$Accuracy = \frac{True_{positive} + True_{negative}}{True_{positive} + True_{negative} + False_{positive} + False_{negative}}$$

- Precision

$$Precision = \frac{True_{positive}}{True_{positive} + False_{positive}}$$

$$AP = \sum_{k=0}^{k=n-1} [Recalls(k) - Recalls(k+1)] * Precisions(k)$$

$$Recalls(n) = 0, Precisions(n) = 1$$

$$n = Number\ of\ thresholds.$$

- Recall

$$Recall = \frac{True_{positive}}{True_{positive} + False_{negative}}$$

- Mean Average Precision

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$

$$AP_k = \ the\ AP\ of\ class\ k$$

$$n = \ the\ number\ of\ classes$$

# Proposed techniques

- Decision Trees
- Gradient Boosting Trees
- Association rules

# Decision Trees

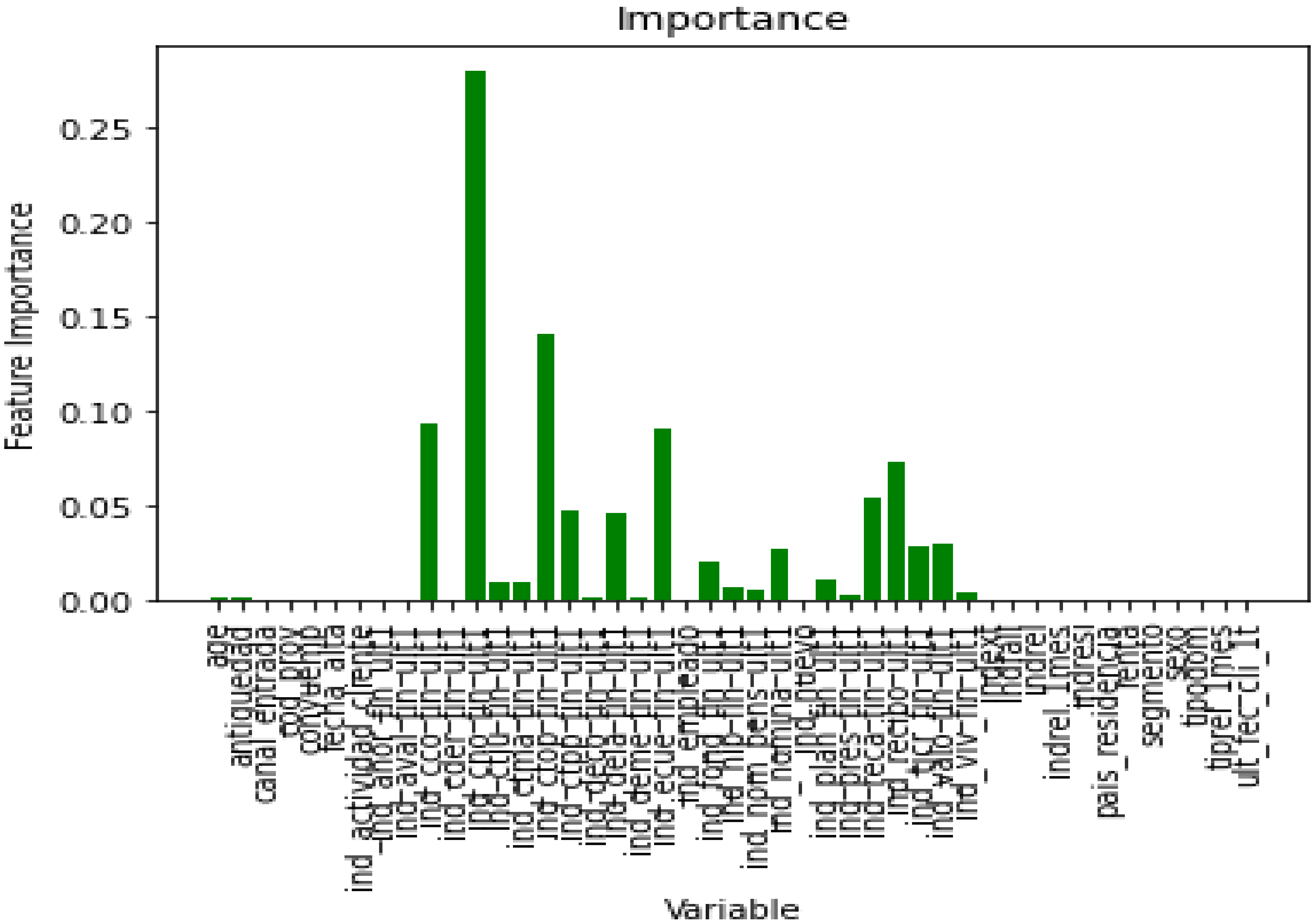
DecisionTreeClassifier(random_state=40, criterion='gini',max_depth=18)

| Accuracy | Precision | MAP@7 |
| --- | --- | --- |
| 92.554% | 97.099% | 0.02339 |

DecisionTreeClassifier(random_state=40, criterion='gini',max_depth=5)

| Accuracy | Precision | MAP@7 |
| --- | --- | --- |
| 92.048% | 96.540% | 0.02339 |

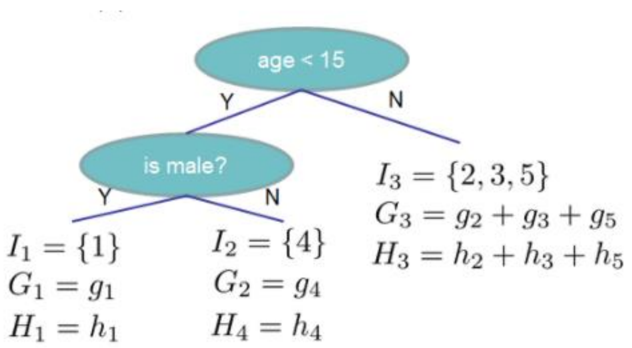
# Decision Trees



# Proposed techniques

- Decision Trees
- Gradient Boosting Trees
- Association rules

# Gradient Boosting Trees




- Prediction is given as: $$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in \mathcal{F}$$
  - where K is the number of trees, f is a function in the functional space F, and F is the set of all possible CARTs
- Additive training: Trees are added at each step of training and the objective function for $t^{th}$ tree:

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)$$

$$\begin{aligned}\text{obj}^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) \\ &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t)\end{aligned}$$

$$\text{obj}^{(t)} = \sum_{i=1}^{n} (y_i - (\hat{y}_i^{(t-1)} + f_t(x_i)))^2 + \sum_{i=1}^{t} \Omega(f_i)$$

$$\sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t)$$

  - l: loss function, Ω: Regularization term, g: gradient of first order, h: gradient of second order

# Gradient Boosting Trees

GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, max_depth=3,random_state=0,subsample=0.5)

| Accuracy | Precision | MAP@7 |
|---|---|---|
| 92.554% | 97.003% | 0.02283 |

GradientBoostingClassifier(n_estimators=50, learning_rate=0.1, max_depth=1,random_state=0,subsample=0.5)

| Accuracy | Precision | MAP@7 |
|---|---|---|
| 92.048% | 96.540% | 0.02312 |

GradientBoostingClassifier(n_estimators=300, learning_rate=0.1, max_depth=1,random_state=0,subsample=0.5)

| Accuracy | Precision | MAP@7 |
|---|---|---|
| 92.706% | 96.844% | 0.02312 |

# Proposed techniques

- Decision Trees
- Gradient Boosting Trees
- Association rules

# Association Rule Mining

- We consider only products purchased to find frequent purchasing patterns
- Rules are learnt using Fpgrowth algorithm
  - fpgrowth(train, min_support=0.00001, use_colnames=False, max_len=None, verbose=0)
- The rules are filtered to get consequents as 1-itemset
- Using learned rules we use confidence as a product scoring metric and predict recommendations for next month

| MAP@7 |
| --- |
| 0.01144 |

# Association Rule Mining

| support | itemsets | Description |
|---|---|---|
| 0.650577 | (ind_cco_fin_ult1) | Current Accounts |
| 0.061361 | (ind_tjcr_fin_ult1) | Credit Card |
| 0.038844 | (ind_ctpp_fin_ult1) | particular Plus Account |
| 0.025175 | (ind_valo_fin_ult1) | Securities |
| 0.164312 | (ind_recibo_ult1) | Direct Debit |

*Rule:* $X \Rightarrow Y$

$$Support = \frac{frq(X,Y)}{N}$$

$$Confidence = \frac{frq(X,Y)}{frq(X)}$$

$$Lift = \frac{Support}{Supp(X) \times Supp(Y)}$$

# Association Rule Mining

$$Rule:\ X \Rightarrow Y$$

$$Support = \frac{frq(X,Y)}{N}$$

$$Confidence = \frac{frq(X,Y)}{frq(X)}$$

$$Lift = \frac{Support}{Supp(X) \times Supp(Y)}$$

| antecedents | consequents | antecedent support | consequent support | support | confidence | lift |
|---|---|---|---|---|---|---|
| (ind_tjcr_fin_ult1, ind_ctju_fin_ult1, ind_cco... | (ind_ecue_fin_ult1) | 0.000010 | 0.085633 | 0.000010 | 1.0 | 11.677795 |
| (ind_tjcr_fin_ult1, ind_ecue_fin_ult1, ind_rec... | (ind_nomina_ult1) | 0.000024 | 0.073545 | 0.000024 | 1.0 | 13.597206 |
| (ind_tjcr_fin_ult1, ind_ecue_fin_ult1, ind_rec... | (ind_nom_pens_ult1) | 0.000020 | 0.078929 | 0.000020 | 1.0 | 12.669620 |
| (ind_plan_fin_ult1, ind_nom_pens_ult1, ind_cco... | (ind_recibo_ult1) | 0.000038 | 0.164312 | 0.000038 | 1.0 | 6.085993 |
| (ind_tjcr_fin_ult1, ind_ecue_fin_ult1, ind_rec... | (ind_recibo_ult1) | 0.000024 | 0.164312 | 0.000024 | 1.0 | 6.085993 |

# Conclusions

- Product purchased features are more important than demographics in recommendation

# Future Work and Improvements

- Since products purchased are most important features there is scope to improve association rules technique
- We should consider markov chains of order > 1 for classification
- Sequential modelling such as Hidden markov model and Recurrent Neural Networks can be explored to learn transition probabilities

Thank You

Question and Comments