A systematic evaluation of regressions and loss functions for the prediction of monetary value in RFM analysis

MSc Research Project
Data Analytics

Shiva Prasad Aruva
Student ID: x22115188

School of Computing
National College of Ireland

Supervisor: Dr Giovani Estrada

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Shiva Prasad Aruva
Student ID:	x22115188
Programme:	Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Dr Giovani Estrada
Submission Due Date:	4/12/2023
Project Title:	A systematic evaluation of regressions and loss functions for the prediction of monetary value in RFM analysis
Word Count:	789
Page Count:	6

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Shiva Prasad Aruva
Date:	4th December 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

A systematic evaluation of regressions and loss functions for the prediction of monetary value in RFM analysis

Shiva Prasad Aruva
x22115188

1 Introduction

This configuration handbook contains Technical specifications and a Description of the hardware and software utilized in the project for Customer Segmentation. Follow the guidelines in this manual to reproduce the results and valuation of regressions and loss functions for the prediction of monetary value in RFM analysis.

2 System Specification

2.1 Hardware Requirements:

Table 1:

Processer	Intel CORE i5 x64
RAM	8GB
DISK Storage	1GB Approx

2.2 Software Requirements:

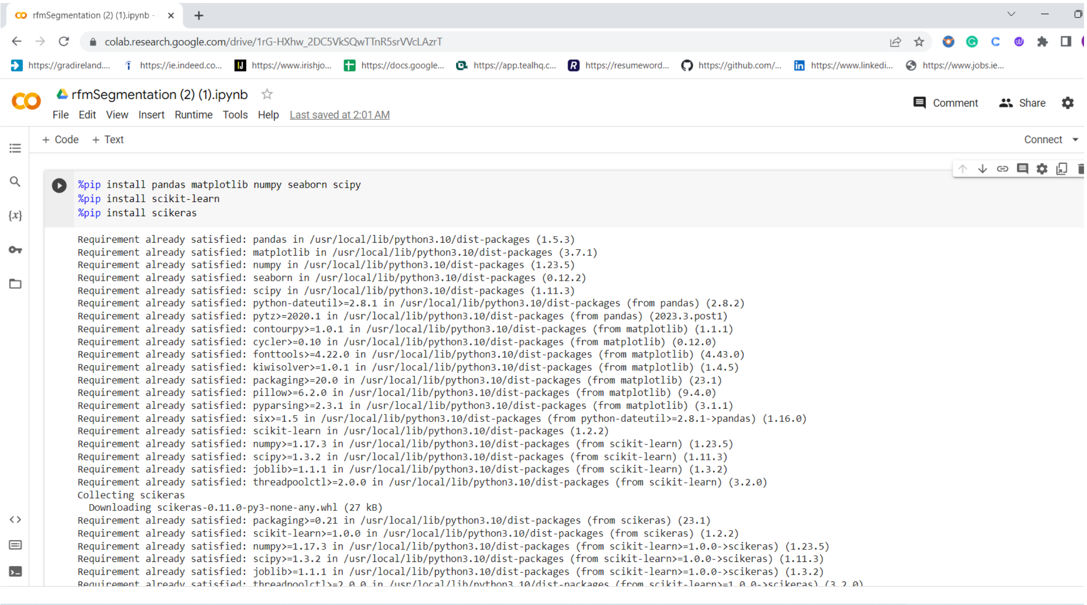
Table 2:

Operating System	Windows 11
Programming Language	Python version 3.10
Web-Broser	Google Chrome
Other Softwares	Google Colaborator, Excel

3 Environment Embedding:

3.1 Google Colaborator Setup

This is the initial stage where we Run the Google Colab. With Colab, anyone can write and run any Python code through a browser, making it particularly useful for data analysis, machine learning. And it has default run-time to Python version 3.10.


rfmSegmentation (2) (1).ipynb
colab.research.google.com/drive/1rG-HXhw_2DC5VkSQwTTnR5srVVcLAzrT
rfmSegmentation (2) (1).ipynb
File Edit View Insert Runtime Tools Help Last saved at 2:01 AM
Comment
Share
+ Code + Text
Connect

```
%pip install pandas matplotlib numpy seaborn scipy
%pip install scikit-learn
%pip install scikeras
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (1.5.3)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (1.23.5)
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (0.12.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (1.11.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.3.post1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.1.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.43.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (23.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.23.5)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.3)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.2.0)
Collecting scikeras
  Downloading scikeras-0.11.0-py3-none-any.whl (27 kB)
Requirement already satisfied: packaging>=0.21 in /usr/local/lib/python3.10/dist-packages (from scikeras) (23.1)
Requirement already satisfied: scikit-learn>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from scikeras) (1.2.2)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.0->scikeras) (1.23.5)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.0->scikeras) (1.11.3)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.0->scikeras) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.0->scikeras) (3.2.0)
```

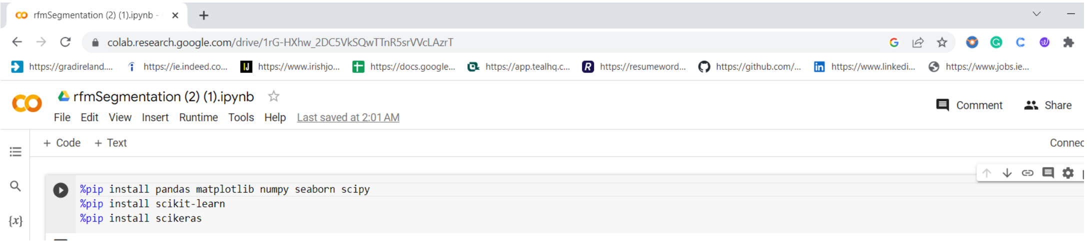
Figure 1: Google Collaborator

3.2 Data Collection

The information can be found for download in CSV format and is taken from the source: `https://archive.ics.uci.edu/dataset/352/online+retail`

3.3 Imported Libraries

These are the list of libraries used for this entire research project 2
3

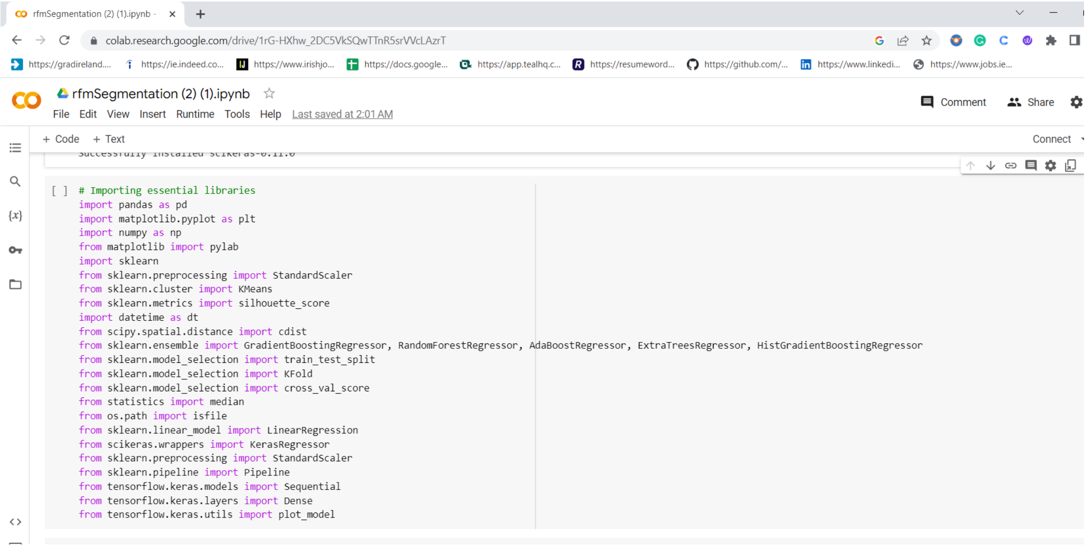

rfmSegmentation (2) (1).ipynb
colab.research.google.com/drive/1rG-HXhw_2DC5VkSQwTTnR5srVVcLAzrT
rfmSegmentation (2) (1).ipynb
File Edit View Insert Runtime Tools Help Last saved at 2:01 AM
Comment
Share
+ Code + Text
Connec

```
%pip install pandas matplotlib numpy seaborn scipy
%pip install scikit-learn
%pip install scikeras
```

Figure 2:

3.4 Data Pre-processing

The pre-processed dataset is uploaded to the Google Collab environment.As seen in the illustration, A selection of the data cleaning procedures is displayed in the figures below 4 , for later usage.



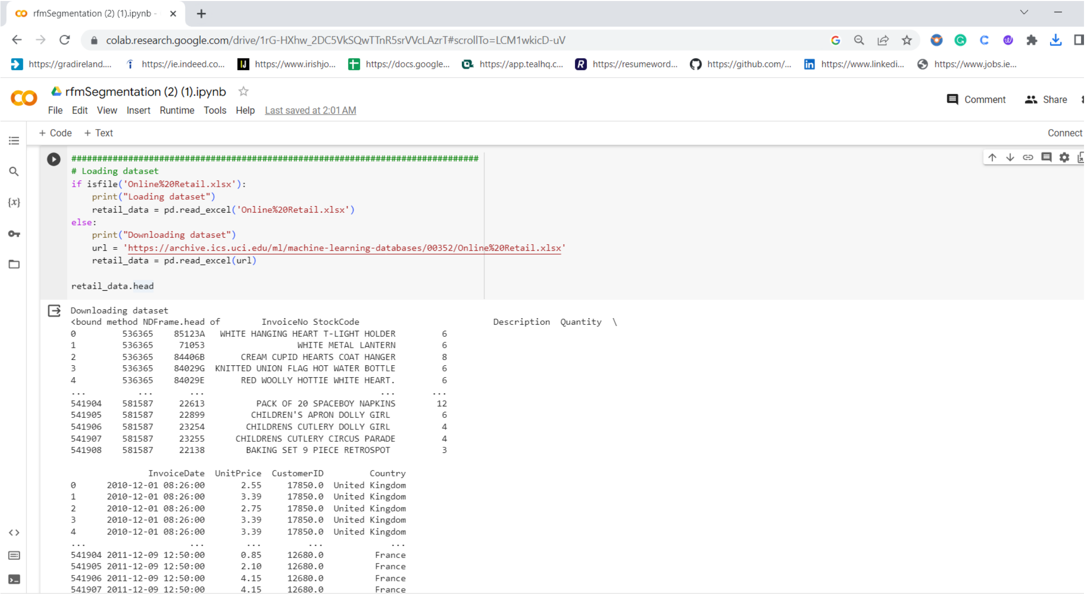
rfmSegmentation (2) (1).ipynb
colab.research.google.com/drive/1rG-HXhw_2DC5VkSQwTTnR5srVVcLAzrT
rfmSegmentation (2) (1).ipynb
File Edit View Insert Runtime Tools Help Last saved at 2:01 AM
Comment
Share
+ Code + Text
Connect

```
Successfully installed scikeras-0.11.0
```

```
# Importing essential libraries
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import pylab
import sklearn
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import datetime as dt
from scipy.spatial.distance import cdist
from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor, AdaBoostRegressor, ExtraTreesRegressor, HistGradientBoostingRegressor
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from statistics import median
from os.path import isfile
from sklearn.linear_model import LinearRegression
from scikeras.wrappers import KerasRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import plot_model
```

Figure 3:

The RFM score is computed 5 for every client in the ”retail data” data frame by this line of code. The reliability, frequency, and monetary worth of a client are measured by the RFM score.



rfmSegmentation (2) (1).ipynb
colab.research.google.com/drive/1rG-HXhw_2DC5VkSQwTTnR5srVVcLAzrT#scrollTo=LCM1wkicD-uV
rfmSegmentation (2) (1).ipynb
File Edit View Insert Runtime Tools Help Last saved at 2:01 AM
Comment
Share
+ Code + Text
Connect

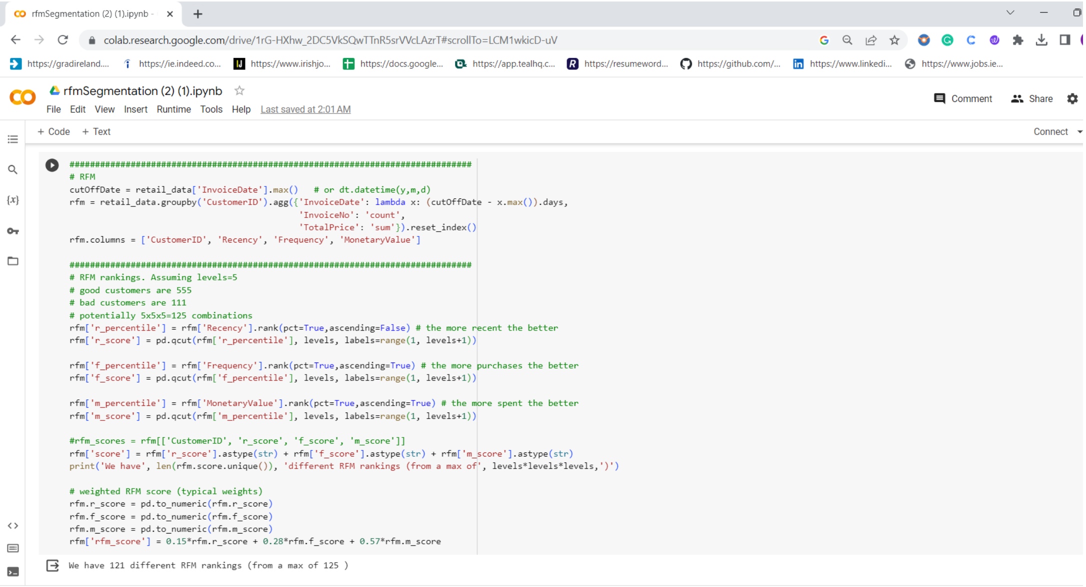
```
################################################################################
# Loading dataset
if isfile('Online%20Retail.xlsx'):
    print("Loading dataset")
    retail_data = pd.read_excel('Online%20Retail.xlsx')
else:
    print("Downloading dataset")
    url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/00352/Online%20Retail.xlsx'
    retail_data = pd.read_excel(url)

retail_data.head
```

```
Downloading dataset
<bound method NDFrame.head of        InvoiceNo StockCode                          Description  Quantity  \
0         536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6
1         536365     71053                  WHITE METAL LANTERN         6
2         536365    84406B       CREAM CUPID HEARTS COAT HANGER         8
3         536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6
4         536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6
...          ...       ...                                  ...       ...
541904    581587     22613          PACK OF 20 SPACEBOY NAPKINS        12
541905    581587     22899         CHILDREN'S APRON DOLLY GIRL          6
541906    581587     23254        CHILDRENS CUTLERY DOLLY GIRL          4
541907    581587     23255      CHILDRENS CUTLERY CIRCUS PARADE         4
541908    581587     22138        BAKING SET 9 PIECE RETROSPOT          3

               InvoiceDate  UnitPrice  CustomerID         Country
0      2010-12-01 08:26:00       2.55     17850.0  United Kingdom
1      2010-12-01 08:26:00       3.39     17850.0  United Kingdom
2      2010-12-01 08:26:00       2.75     17850.0  United Kingdom
3      2010-12-01 08:26:00       3.39     17850.0  United Kingdom
4      2010-12-01 08:26:00       3.39     17850.0  United Kingdom
...                    ...        ...         ...             ...
541904 2011-12-09 12:50:00       0.85     12680.0          France
541905 2011-12-09 12:50:00       2.10     12680.0          France
541906 2011-12-09 12:50:00       4.15     12680.0          France
541907 2011-12-09 12:50:00       4.15     12680.0          France
```

Figure 4:

3.5 Plotting Elbow meathod

Here We Plot the inertia against the number of clusters is a technique known as the elbow method 6, which helps determine the ideal number of clusters. The within-cluster variation is measured by the inertia, which gets smaller as the number of clusters rises. The elbow’s location on the plot indicates the ideal number of clusters.



```
################################################################################
# RFM
cutOffDate = retail_data['InvoiceDate'].max()   # or dt.datetime(y,m,d)
rfm = retail_data.groupby('CustomerID').agg({'InvoiceDate': lambda x: (cutOffDate - x.max()).days,
                                              'InvoiceNo': 'count',
                                              'TotalPrice': 'sum'}).reset_index()
rfm.columns = ['CustomerID', 'Recency', 'Frequency', 'MonetaryValue']

################################################################################
# RFM rankings. Assuming levels=5
# good customers are 555
# bad customers are 111
# potentially 5x5x5=125 combinations
rfm['r_percentile'] = rfm['Recency'].rank(pct=True,ascending=False) # the more recent the better
rfm['r_score'] = pd.qcut(rfm['r_percentile'], levels, labels=range(1, levels+1))

rfm['f_percentile'] = rfm['Frequency'].rank(pct=True,ascending=True) # the more purchases the better
rfm['f_score'] = pd.qcut(rfm['f_percentile'], levels, labels=range(1, levels+1))

rfm['m_percentile'] = rfm['MonetaryValue'].rank(pct=True,ascending=True) # the more spent the better
rfm['m_score'] = pd.qcut(rfm['m_percentile'], levels, labels=range(1, levels+1))

#rfm_scores = rfm[['CustomerID', 'r_score', 'f_score', 'm_score']]
rfm['score'] = rfm['r_score'].astype(str) + rfm['f_score'].astype(str) + rfm['m_score'].astype(str)
print('We have', len(rfm.score.unique()), 'different RFM rankings (from a max of', levels*levels*levels,')')

# weighted RFM score (typical weights)
rfm.r_score = pd.to_numeric(rfm.r_score)
rfm.f_score = pd.to_numeric(rfm.f_score)
rfm.m_score = pd.to_numeric(rfm.m_score)
rfm['rfm_score'] = 0.15*rfm.r_score + 0.28*rfm.f_score + 0.57*rfm.m_score
```

```
We have 121 different RFM rankings (from a max of 125 )
```

Figure 5:

For k-values between 2 and 40, you are charting the inertia versus the number of clusters in your code. This is the ideal amount of clusters for your data, as the elbow seems to be around k=5.

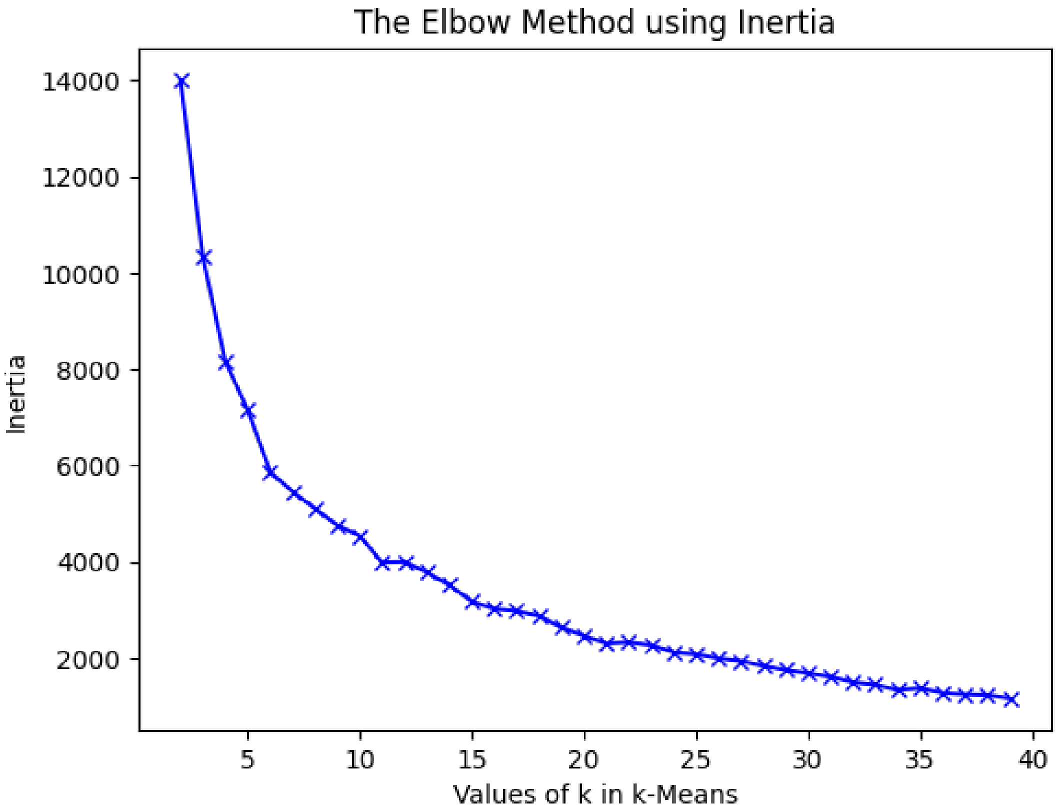
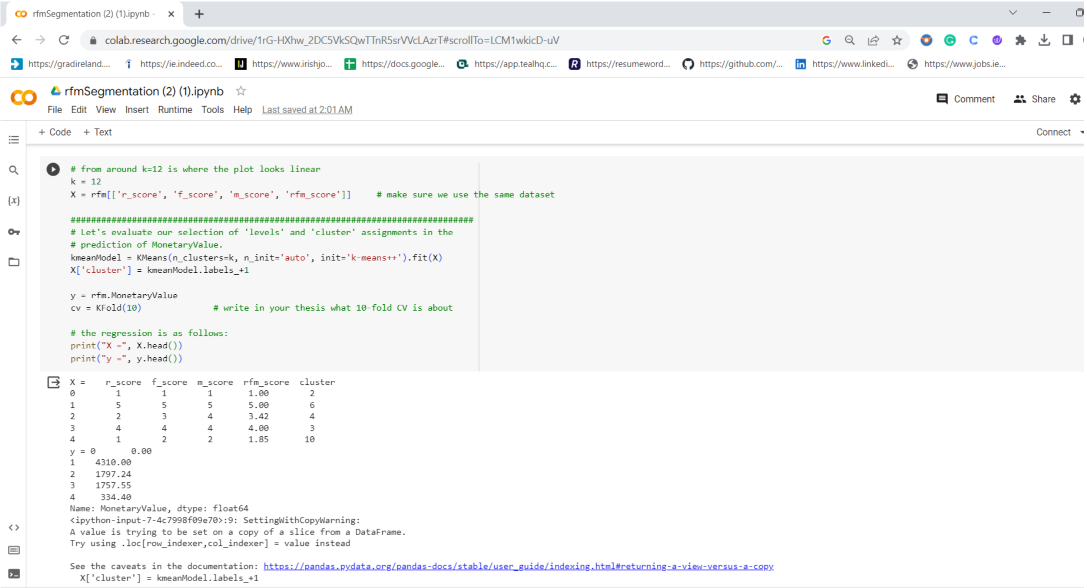


Figure 6:

3.6 Evaluating the performance

The figure 7 below evaluates the performance of k-means clustering in predicting the Monetary Value variable.



```
# from around k=12 is where the plot looks linear
k = 12
X = rfm[['r_score', 'f_score', 'm_score', 'rfm_score']]     # make sure we use the same dataset

###############################################################################
# Let's evaluate our selection of 'levels' and 'cluster' assignments in the
# prediction of MonetaryValue.
kmeanModel = KMeans(n_clusters=k, n_init='auto', init='k-means++').fit(X)
X['cluster'] = kmeanModel.labels_+1

y = rfm.MonetaryValue
cv = KFold(10)              # write in your thesis what 10-fold CV is about

# the regression is as follows:
print("X =", X.head())
print("y =", y.head())
```

```
X =    r_score  f_score  m_score  rfm_score  cluster
0        1        1        1       1.00        2
1        5        5        5       5.00        6
2        2        3        4       3.42        4
3        4        4        4       4.00        3
4        1        2        2       1.85       10
y = 0       0.00
1    4310.00
2    1797.24
3    1757.55
4     334.40
Name: MonetaryValue, dtype: float64
<ipython-input-7-4c7998f09e70>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X['cluster'] = kmeanModel.labels_+1
```

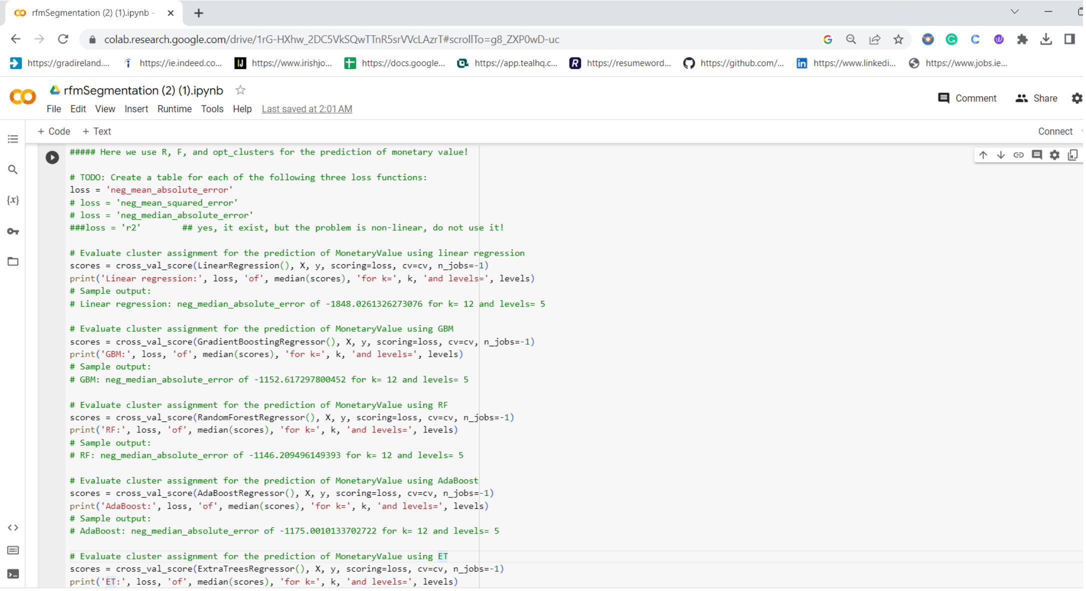
Figure 7:

4 Performance of various machine learning algorithms using Loss Functions

Here The performance of various machine learning algorithms 8 is evaluated using the Loss functions for the negative mean absolute error, negative mean squared error, and negative median absolute error. In this, we Always consider the Lower values are better suggested fit for the model.

5 Defining Neural Network For Regression

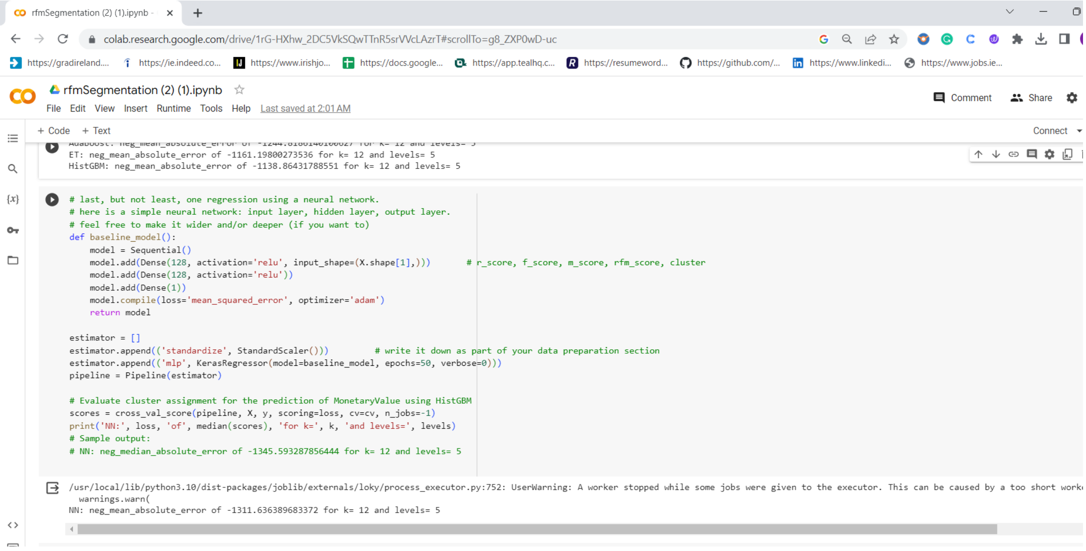
In this below figure 9 we Define a neural network model for regression and evaluate its performance using cross-validation.The neural network's efficiency can be assessed using the obtained scores.



rfmSegmentation (2) (1).ipynb
colab.research.google.com/drive/1rG-HXhw_2DC5VkSQwTTnR5srVVcLAzrT#scrollTo=g8_ZXP0wD-uc
File
Edit
View
Insert
Runtime
Tools
Help
Last saved at 2:01 AM
Comment
Share
+ Code
+ Text
Connect

```
##### Here we use R, F, and opt_clusters for the prediction of monetary value!

# TODO: Create a table for each of the following three loss functions:
loss = 'neg_mean_absolute_error'
# loss = 'neg_mean_squared_error'
# loss = 'neg_median_absolute_error'
###loss = 'r2'        ## yes, it exist, but the problem is non-linear, do not use it!

# Evaluate cluster assignment for the prediction of MonetaryValue using linear regression
scores = cross_val_score(LinearRegression(), X, y, scoring=loss, cv=cv, n_jobs=-1)
print('Linear regression:', loss, 'of', median(scores), 'for k=', k, 'and levels=', levels)
# Sample output:
# Linear regression: neg_median_absolute_error of -1848.0261326273076 for k= 12 and levels= 5

# Evaluate cluster assignment for the prediction of MonetaryValue using GBM
scores = cross_val_score(GradientBoostingRegressor(), X, y, scoring=loss, cv=cv, n_jobs=-1)
print('GBM:', loss, 'of', median(scores), 'for k=', k, 'and levels=', levels)
# Sample output:
# GBM: neg_median_absolute_error of -1152.617297800452 for k= 12 and levels= 5

# Evaluate cluster assignment for the prediction of MonetaryValue using RF
scores = cross_val_score(RandomForestRegressor(), X, y, scoring=loss, cv=cv, n_jobs=-1)
print('RF:', loss, 'of', median(scores), 'for k=', k, 'and levels=', levels)
# Sample output:
# RF: neg_median_absolute_error of -1146.209496149393 for k= 12 and levels= 5

# Evaluate cluster assignment for the prediction of MonetaryValue using AdaBoost
scores = cross_val_score(AdaBoostRegressor(), X, y, scoring=loss, cv=cv, n_jobs=-1)
print('AdaBoost:', loss, 'of', median(scores), 'for k=', k, 'and levels=', levels)
# Sample output:
# AdaBoost: neg_median_absolute_error of -1175.0010133702722 for k= 12 and levels= 5

# Evaluate cluster assignment for the prediction of MonetaryValue using ET
scores = cross_val_score(ExtraTreesRegressor(), X, y, scoring=loss, cv=cv, n_jobs=-1)
print('ET:', loss, 'of', median(scores), 'for k=', k, 'and levels=', levels)
```

Figure 8:



rfmSegmentation (2) (1).ipynb
colab.research.google.com/drive/1rG-HXhw_2DC5VkSQwTTnR5srVVcLAzrT#scrollTo=g8_ZXP0wD-uc
File
Edit
View
Insert
Runtime
Tools
Help
Last saved at 2:01 AM
Comment
Share
+ Code
+ Text
Connect

```
AdaBoost: neg_mean_absolute_error of -1244.8186140166027 for k= 12 and levels= 5
ET: neg_mean_absolute_error of -1161.19800273536 for k= 12 and levels= 5
HistGBM: neg_mean_absolute_error of -1138.86431788551 for k= 12 and levels= 5
```

```
# last, but not least, one regression using a neural network.
# here is a simple neural network: input layer, hidden layer, output layer.
# feel free to make it wider and/or deeper (if you want to)
def baseline_model():
    model = Sequential()
    model.add(Dense(128, activation='relu', input_shape=(X.shape[1],)))       # r_score, f_score, m_score, rfm_score, cluster
    model.add(Dense(128, activation='relu'))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
    return model

estimator = []
estimator.append(('standardize', StandardScaler()))         # write it down as part of your data preparation section
estimator.append(('mlp', KerasRegressor(model=baseline_model, epochs=50, verbose=0)))
pipeline = Pipeline(estimator)

# Evaluate cluster assignment for the prediction of MonetaryValue using HistGBM
scores = cross_val_score(pipeline, X, y, scoring=loss, cv=cv, n_jobs=-1)
print('NN:', loss, 'of', median(scores), 'for k=', k, 'and levels=', levels)
# Sample output:
# NN: neg_median_absolute_error of -1345.593287856444 for k= 12 and levels= 5
```

```
/usr/local/lib/python3.10/dist-packages/joblib/externals/loky/process_executor.py:752: UserWarning: A worker stopped while some jobs were given to the executor. This can be caused by a too short work
  warnings.warn(
NN: neg_mean_absolute_error of -1311.636389683372 for k= 12 and levels= 5
```

Figure 9: