Data Cleansing

Objective: Clean Up Unnecessary Data Before Exploring The Data Source

Number of Rows: 12403
Number of Fields: 23
Number of Customers: 100

Understand the Field Data

```
status                 object
card_present_flag     float64
account                object
long_lat               object
txn_description        object
merchant_id            object
first_name             object
balance               float64
date                   object
gender                 object
age                     int64
merchant_suburb        object
merchant_state         object
extraction             object
amount                float64
transaction_id         object
customer_id            object
merchant_long_lat      object
movement               object
dtype: object
```

Data schema

Validate Missing Data on Each Field

```
Missing Percentage is status
card_present_flag     35.92
bpay_biller_code      92.65
account                0.00
currency               0.00
long_lat               0.00
txn_description        0.00
merchant_id           35.92
merchant_code         92.67
first_name             0.00
balance                0.00
date                   0.00
gender                 0.00
age                    0.00
merchant_suburb       35.92
merchant_state        35.92
extraction             0.00
amount                 0.00
transaction_id         0.00
country                0.00
customer_id            0.00
merchant_long_lat     35.92
movement               0.00
```

Jupyter Noteebok (AWS SageMaker)

Eliminate 4 Unnecessary Columns

- Currency

```
AUD    12043
Name: currency, dtype: int64
```

```
df.currency.unique()
```

```
array(['AUD'], dtype=object)
```

- Country
- Bpay Biller Code

```
df.bpay_biller_code.value_counts()
```

```
0                                          883
 LAND WATER & PLANNING East Melbourne        1
 THE DISCOUNT CHEMIST GROUP                  1
Name: bpay_biller_code, dtype: int64
```

```
# 2 values only represent 0.016%, which is not ideal.
```

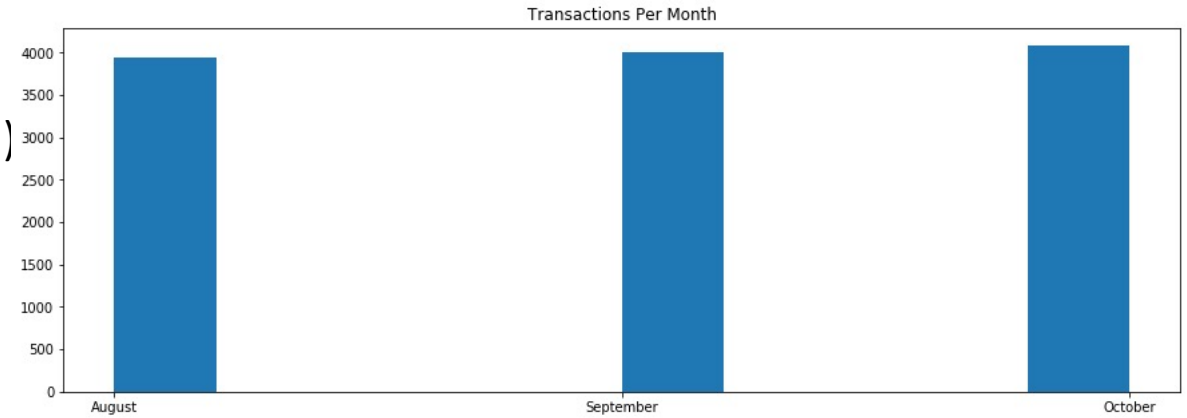
- Merchant Code

Data Discovery

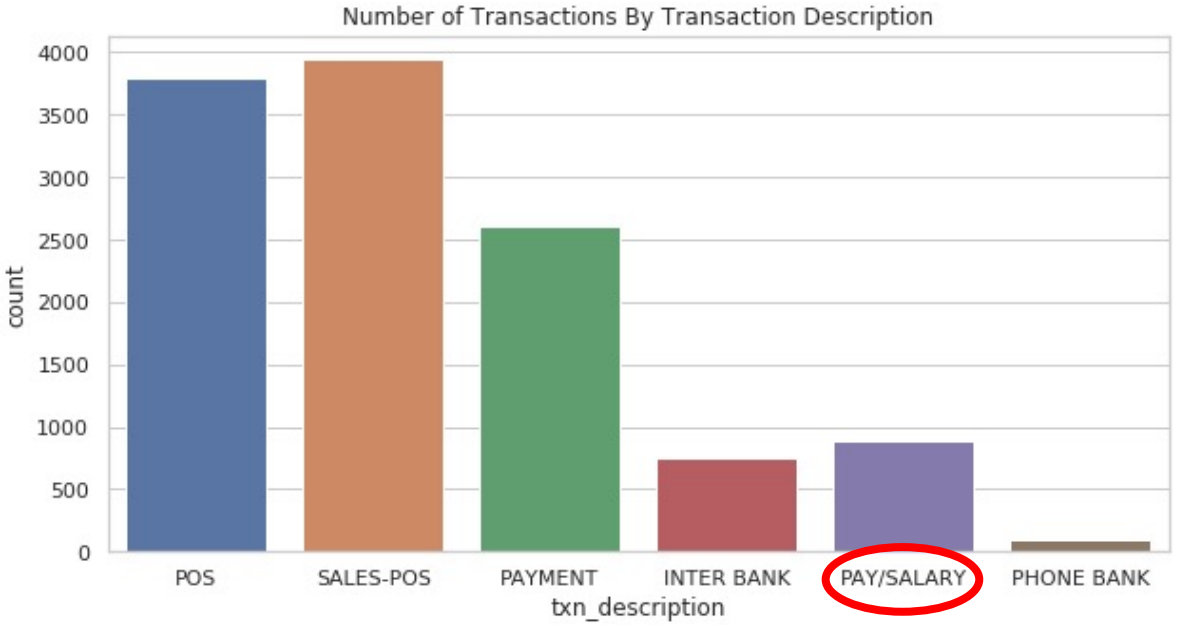
Objective: Understand The Basic Data Set

All Transactions in August 2018 - October 2018 (Seasonal)

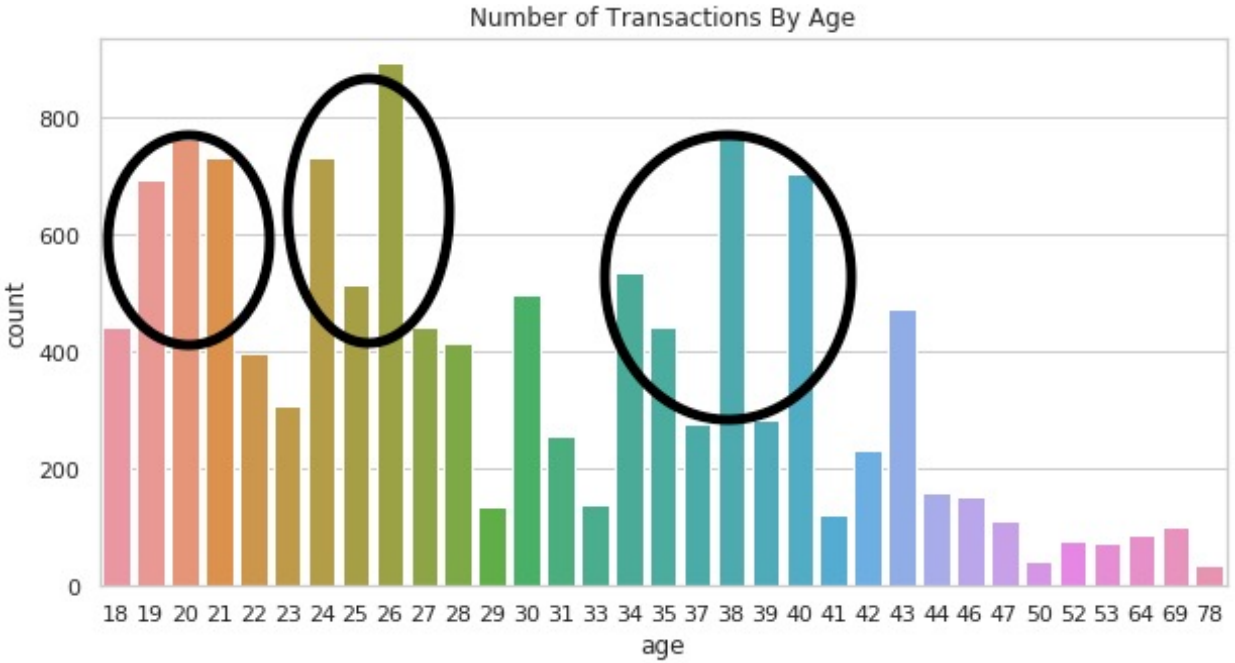
# of Transactions By Gender:
Male: 6285 (9% Higher) Female: 5758



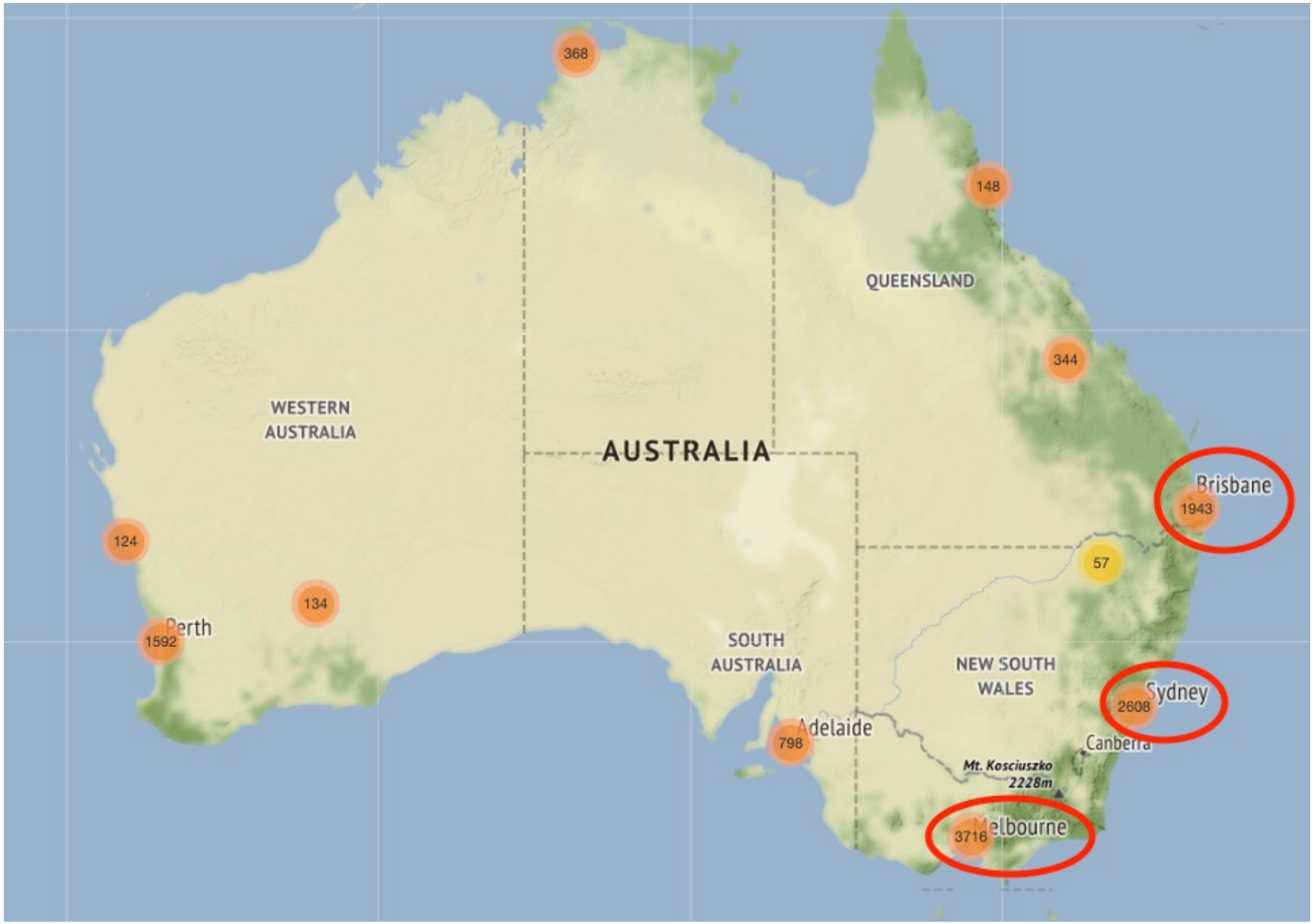
August: 3943 September: 4013 October: 4087



7.33% is Salary Payment (100% Credit Transactions)



Data Discovery



Data Insights

Objective: Analyze Data Patterns

Customers Receive Salary on Different Cycle

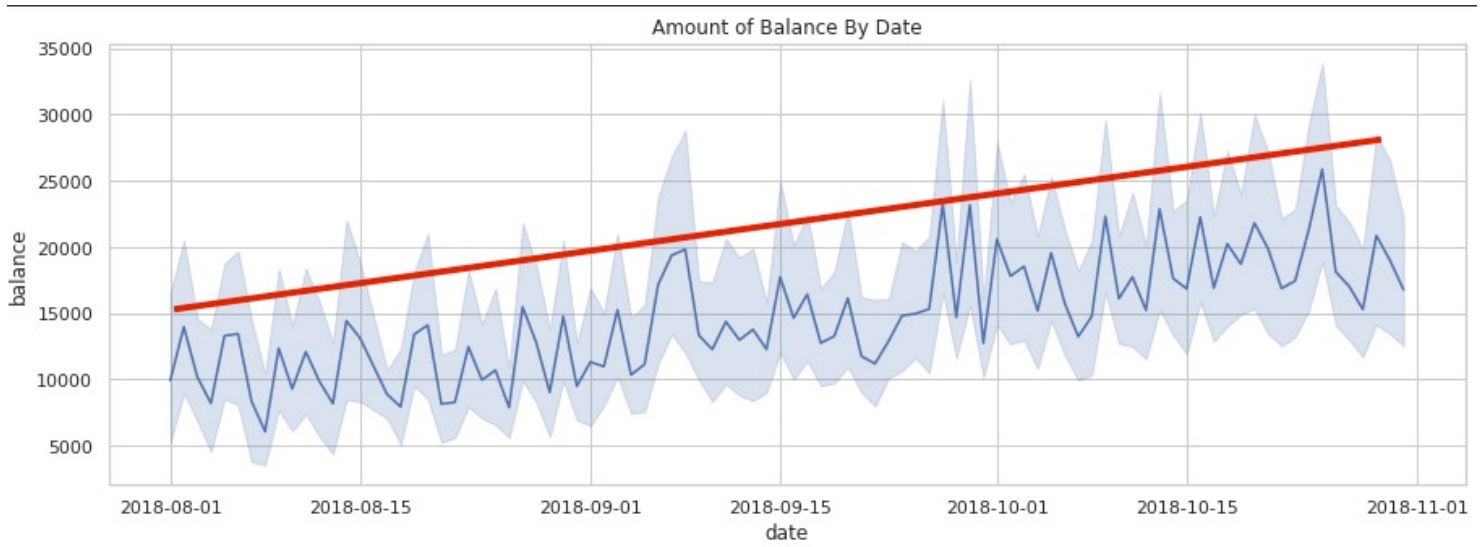
```
SELECT customer_id, date, age, amount, gender
FROM demo_db.dataset_demo
JOIN (SELECT distinct amount amount3
      FROM demo_db.dataset_demo
      WHERE movement = 'credit'
      ORDER BY amount DESC
      LIMIT 2, 1) sq
ON amount <= amount3
ORDER by amount DESC
```

100% 21:8

```
    WHERE movement = 'credit'
    ORDER BY amount DESC
    LIMIT 2, 1) sq
ON amount <= amount3
ORDER by amount DESC
```

customer_id	date	age	amount	gender
CUS-51506836	8/7/2018	24	6024.49	M
CUS-51506836	9/7/2018	24	6024.49	M
CUS-2376382098	8/15/2018	39	5103.51	F
CUS-2376382098	9/14/2018	39	5103.51	F
CUS-2376382098	9/14/2018	39	5103.51	F
CUS-2376382098	10/15/2018	39	5103.51	F
CUS-2376382098	10/15/2018	39	5103.51	F
CUS-497688347	8/23/2018	30	4910.9	F
CUS-497688347	10/23/2018	30	4910.9	F
CUS-1739931018	9/26/2018	27	4863.62	F
CUS-1739931018	10/26/2018	27	4863.62	F
CUS-261674136	8/6/2018	29	4405.3	M
CUS-261674136	8/20/2018	29	4405.3	M
CUS-261674136	9/3/2018	29	4405.3	M
CUS-261674136	9/17/2018	29	4405.3	M
CUS-261674136	10/1/2018	29	4405.3	M
CUS-261674136	10/15/2018	29	4405.3	M
CUS-261674136	10/29/2018	29	4405.3	M

From MySQL (AWS RDS)



Balance Increase from AUD 150,000 to 250,000

Customers Have Been Saving (67 % Increase) Within 3 Months

Annual Salary Calculation Is Required

Data Heatmap

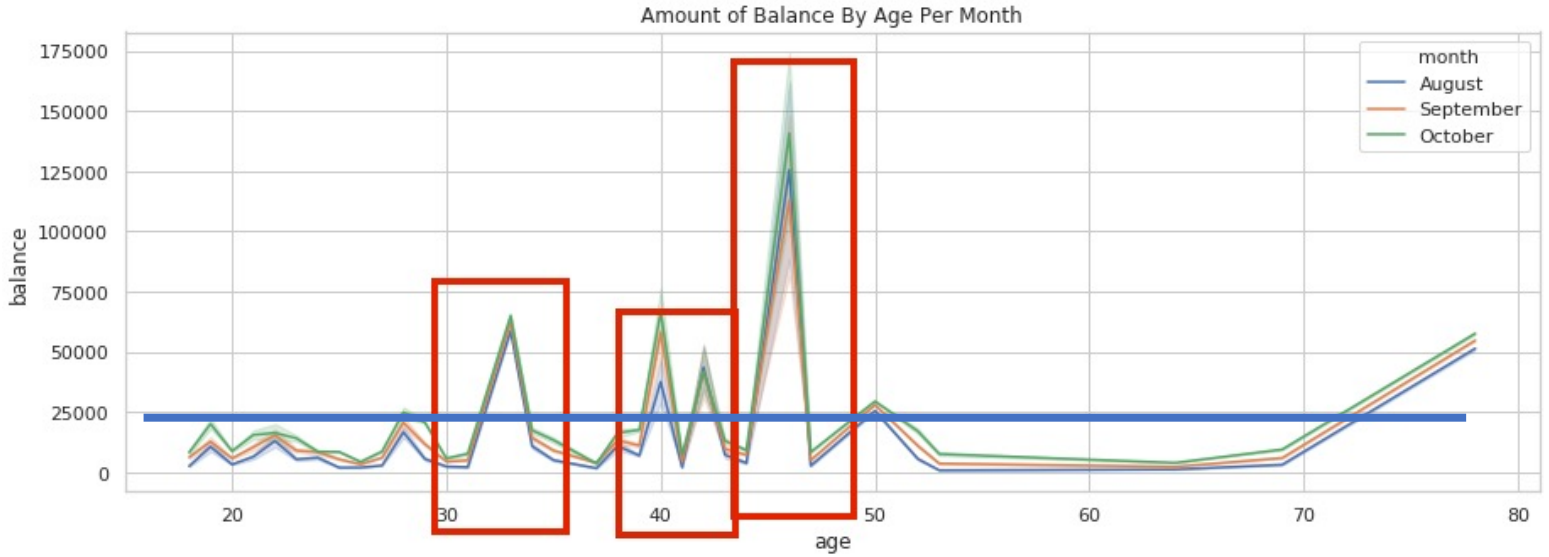
```python
# iterate customers based on salary transaction type

for count, cs_id in enumerate(customers):
    df_cus = df[(df.customer_id == cs_id) & (df.txn_description == 'PAY/SALARY')]

    # calculate annual salary
    pay_period = pd.to_datetime(df_cus.date).diff().mean() / pd.Timedelta('86400s')
    pay_amount = df_cus.amount.mean()
    daily_rate = pay_amount / pay_period
    annual_rate = daily_rate * 365

    # get age
    age = df_cus.age.mean()
    # get gender
    gender = df_cus["gender"].mode()[0]

    # store the info
    df_cus_profile.loc[count, ["customer_id", "annual salary", "age", "gender" ]] = [id, annual_rate, age, gender]
```

When Transaction's Type is Salary/Payment,
Fetch Customers' Salary
Age
Gender

```python
# also iterate customers with nonreq. where salary needs to be included

for count, cs_id in enumerate(customers):
  df_cus = df[(df.customer_id == cs_id)]

  # rename to state
  state = df_cus["merchant_state"].mode()[0]

  # calculate the avg balance amount
  avg_bal_amount = df_cus["balance"].mean()

  # calculate the max balance amount
  max_bal_amount = df_cus["balance"].max()

  # calculate the avg transaction amount during a certain time of period
  avg_trans_amount = df_cus["amount"].mean()

  # calculate the max trans amount
  max_trans_amount = df_cus["amount"].max()


  # store the info
  df_cus_profile.loc[count, ["state", "avg_bal_amount", "max_bal_amount", "avg_transaction_amount",
                             "max_transaction_amount"]] = [state,
                                                           avg_bal_amount, max_bal_amount,
                                                           avg_trans_amount, max_trans_amount]
```

Fetch Customers' Transaction of
State
Avg./Max. Balance Amount
Avg./Max. Transaction Amount

Data Insights



From Jupyter Notebook (AWS SageMaker)

Age	Total Salary	Total Spending	% of Spending
10 to 19	$175548	$49649.279	28.28 4
20 to 29	$635198.39 1	$230161.259	36.23 2
30 to 39	$449660.8 3	$177494.85	39.47 1
40 to 49	$335513.429	$107910.359	32.16 3
50 to 59	$49175.43 2	$12859.42	26.15
60 to 69	$22091.679	$7175.96	32.48
70 to 79	$9389.039	$1456.21	15.51

Age of 10 - 30 Spends > Age of 40 - 70

Total Amount of Balance By Gender

```
(df[df.txn_description !="PAY/SALARY"].groupby(["gender"]).sum())
```

gender	card_present_flag	balance	age	amount
F	3102.0	64534780.86	161891	266666.40
M	3092.0	98205401.65	177598	320040.95

Men Have 20% Higher Balance

Total Amount of Salary By Gender

```
(df[df.txn_description =="PAY/SALARY"].groupby(["gender"]).sum())
```

gender	card_present_flag	balance	age	amount
F	0.0	4913956.70	13502	703656.23
M	0.0	9428487.84	15312	972920.62

Men Have 38% Higher Salary

At This Stage, Gender Has a Low Impact Data Field

Data Heatmap (Correlation Heatmap)

A Visual Exploratory Data Analysis Tool

Objective: Encounter Relational Variables From A Visual Approach

Range from 0 to 1
Closer to 1 Means Higher Correlation

High Correlation: Annual Salary x Max. Trans. Amount
Reason: Highest Amount (Numeric) is Salary

This Impacts Avg. Transaction Amount As Well

0.58 Has A Medium Correlation

Low Correlation: Age and Avg. (Max.) Balance Amount
Reason: Different % of Spending By Age Range

0.23 Has A Low Correlation

Data Validation

So What Now?

More Data Modelling Required For Data Examination

Reason: I. Mitigate Analytical Error (Risk)

II. Reduce Time & Cost Against Scope and Resource From Re-Cycling Previous Phases

Objective: Constant Validation Required For Accurate Solution

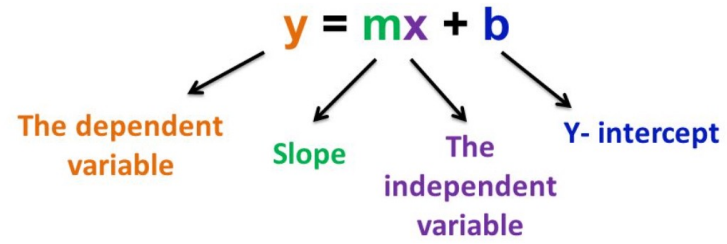
Approach: Use Predictive Analysis > Analyze historical data to make an accurate guess of an unknown subject

Linear Regression

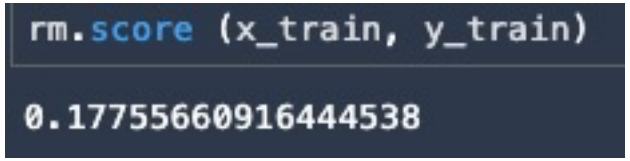
A Statistical Tool to Encounter The Relationship Between The Two Variables

Uses Algorithms in Machine Learning To Test and Predict The Outcome

Steps To Take: Build -> Train (80%) -> Test (20%) The Model

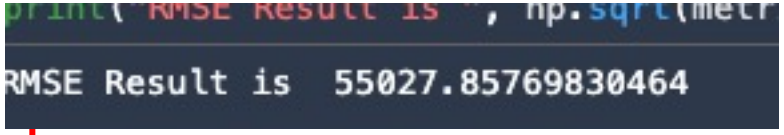


Y: Avg. Balance Amount
X: Age, Annual Salary, Avg. Trans. Amount

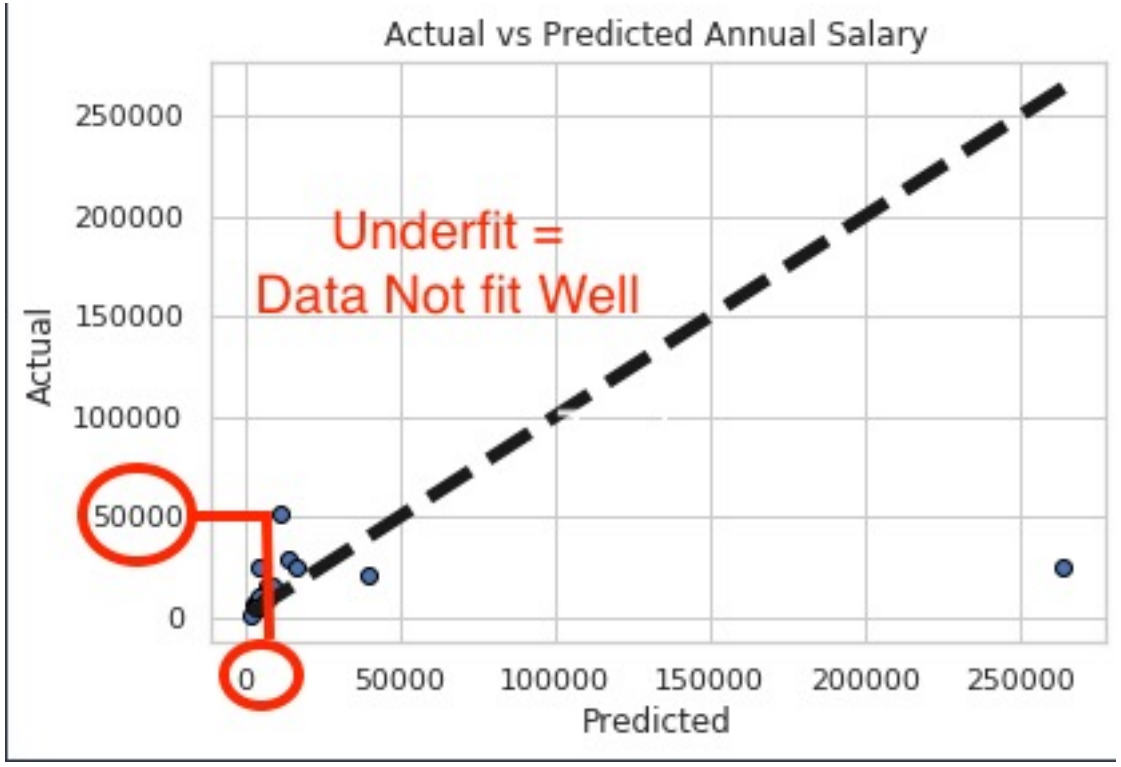


R^2 (Coefficient Determination = Measurement of How Close The Data Fits (How Well X Variable Explains Y Variable)

It Ranges From 0 to 1 (Scaled)
Closer to 1 Means Higher Correlation



Root Means Square Error = Unscaled Measurement (Square Root of Residual – Measure How Far From The Regression Line The Data Points Are)



X Axis - Making Model To Predict Data

Lower the RMSE, The Better The Model Is – 55027 Has Performed Poorly

Decision Tree Regression

A Tool to Capture Data and Divide into Groups to find the Values Relationship

More used to capture Non-Linear Data Shape (opposite to Linear Regression)

Steps To Take: Build -> Train (80%) -> Test (20%) The Model

Y: Avg. Balance Amount
X: Age, Annual Salary, Avg. Trans. Amount

```
model.score(x_train, y_train)
```

```
0.6563793326807585
```

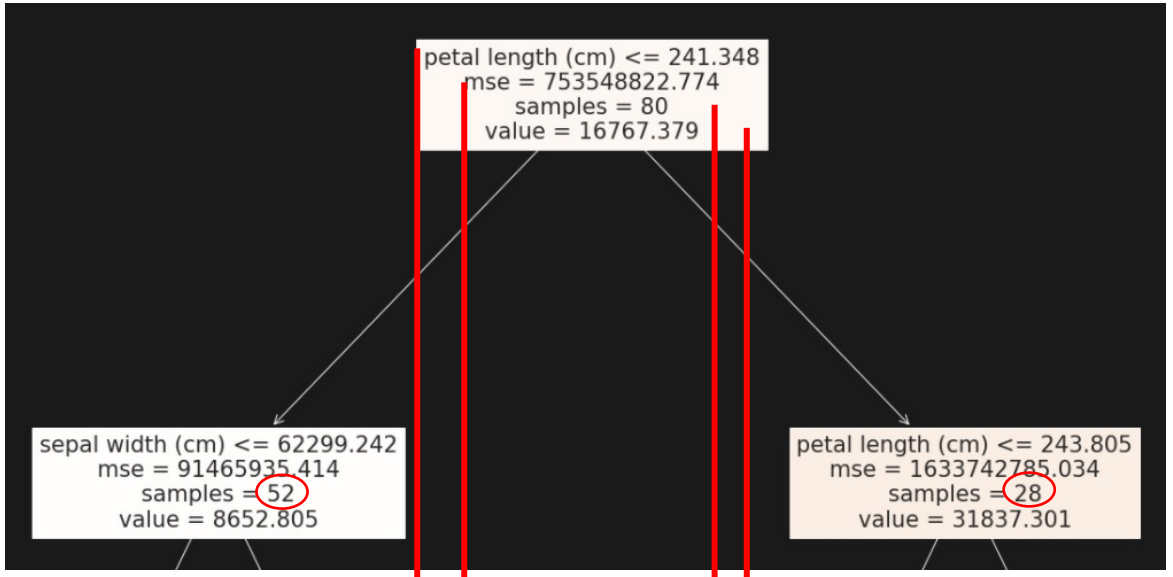
Measurement of how close the data fits
0.65 is an OK measurement

```
print("RMSE Result is ", np.sqrt(metrics.me
```

```
RMSE Result is  54387.679192649055
```

Root Means Square Error = Unscaled Measurement

RMSE is 54387, A Similar Result From Linear (Poorly Performed)



Decision Tree Regression Model

Iris (Dataset Pattern Recognition)
Data Measurement To Design Model

√ (Square Root) of RMSE =
Measures How Closely the Data Fits

Each Row Is A Sample

Mean Value of Y-Axis

Conclusion

Conducted Customers' Spending Pattern Between Aug. 18 – Oct. 18 - A Seasonal Transactional Dataset

Encountered Customers Have Been Building Savings (67 % Increase of Savings)

Group of Age 10 - 30 Have Been More Active On Spending Than The Age of 40 - 70

Business Recommendation: Target Group of Age of 40 - 70 To Be More Proactive with Number of Transactions

Data Recommendation: Inadequate Analysis -> Insufficient Data

Get and Analyze Customers' Transactional Dataset of 2020 For Further Understanding About The Trend

Attempt to Fetch More Data Fields Ex: (Bank Loan – 'Accepted', 'Rejected')

Yes

Categorical Model

Higher Chance to Understand Customers' Trend of Spending

No

Classification Model

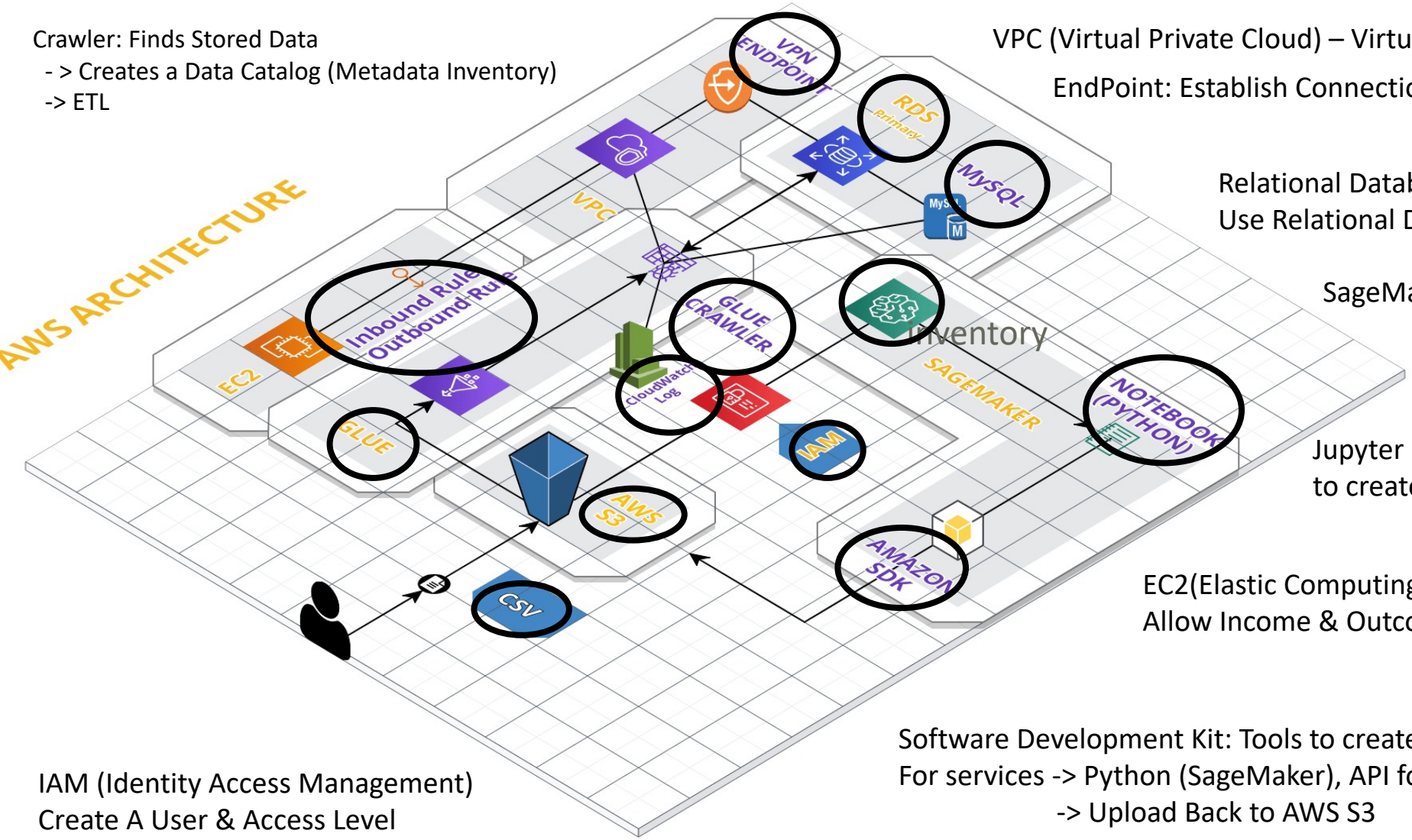
Grouping Data Set Into Classes to Figure Out Patterns For a Particular Event

Ex: Segment Customers Based on Purchasing History For Targeted Marketing (i.e Cashback/Rewards if Spent X Amount)

AWS Architecture

Glue: Serverless Data Integration Service (ETL)

Crawler: Finds Stored Data
- > Creates a Data Catalog (Metadata Inventory)
-> ETL



VPC (Virtual Private Cloud) – Virtual Network (Network Layer) for EC2

EndPoint: Establish Connection with AWS Services

Relational Database Service:
Use Relational Database In AWS

SageMaker: Machine Learning Platform in AWS

Jupyter Notebook: Open Source Web App
to create and share codes/visualization

EC2(Elastic Computing) – Web Service Providing Virtual Server
Allow Income & Outcome Traffic between Crawler and SQL

Software Development Kit: Tools to create an application
For services -> Python (SageMaker), API for AWS Infrastructure
-> Upload Back to AWS S3

IAM (Identity Access Management)
Create A User & Access Level