Prediction who possible Defaulters are for Loans Product, Based on Big-data analysis

#Big-data #MapReduce #ML

Dayoung Kang, Jaehyeon Kim, Subin Seo

What is Big-data ?

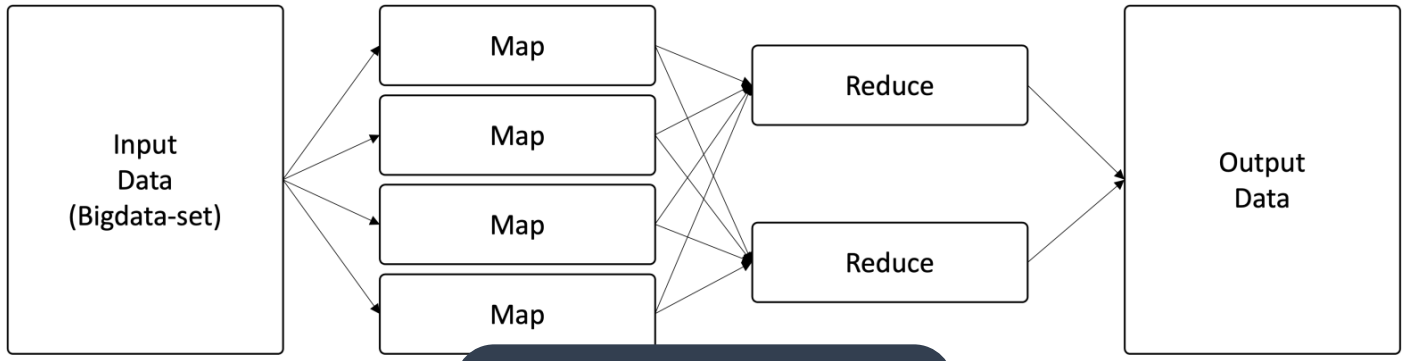
: Any data set contains large volumes of information and complex data is called Big Data (BD). It has 4 characteristics. (4V's) [1]

- **Volume** which is the quantity of data.
- **Velocity** is the speed of the data that during handling and generating.
- **Variety** refers to the range of data types and sources.
- **Veracity** is related to the truth of data which is important for precision in analysis.
- **+ Value** is the importance of the data importance and this is a very significant feature in BD6.

➡ BD is unlike traditional data, so it requires special processing to manage it.

How to process Big-data ?

: apply for MapReduce to process Big Data in parallel on multiple node.



Step1. Map

- Split input data to number of slices
- Apply specific function to each to generate intermediate results

Step2. Reduce

- Combine the intermediate results to make the final result.

Resilient Distributed Dataset(RDD)

: a read-only collection of objects partitioned across a set of machines. [2]

Processing Steps

1. **Generated from the file** – from Shared file systems
2. **Parallelizing Scala collections** – split number of slices ---> nodes
3. **Transformation RDD** – convert type of component
4. **Changing persistence** – Cache & Save

* Cache: keep RDD in memory and make them reusable
* Save: computes RDD and saves to distributed file system
---> making available to subsequent operations

Materials


1
Pre-processing
2
MapReduce
functional programming
3
Classification

About Data

Data Reference : https://www.kaggle.com/datasets/subhamjain/loan-prediction-based-on-customer-behavior?resource=download&select=Sample+Prediction+Dataset.csv [3]

```
train .head(10)
```

	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profession	CITY	STATE	CURRENT_JOB_YRS	CURRENT_HOUSE_YRS	Risk_Flag
0	1	1303834	23	3	single	rented	no	Mechanical_engineer	Rewa	Madhya_Pradesh	3	13	0
1	2	7574516	40	10	single	rented	no	Software_Developer	Parbhani	Maharashtra	9	13	0
2	3	3991815	66	4	married	rented	no	Technical_writer	Alappuzha	Kerala	4	10	0
3	4	6256451	41	2	single	rented	yes	Software_Developer	Bhubaneswar	Odisha	2	12	1
4	5	5768871	47	11	single	rented	no	Civil_servant	Tiruchirappalli[10]	Tamil_Nadu	3	14	1
5	6	6915937	64	0	single	rented	no	Civil_servant	Jalgaon	Maharashtra	0	12	0
6	7	3954973	58	14	married	rented	no	Librarian	Tiruppur	Tamil_Nadu	8	12	0
7	8	1706172	33	2	single	rented	no	Economist	Jamnagar	Gujarat	2	14	0
8	9	7566849	24	17	single	rented	yes	Flight_attendant	Kota[6]	Rajasthan	11	11	0
9	10	8964846	23	12	single	rented	no	Architect	Karimnagar	Telangana	5	13	0

Independent variables (X)

Dependent variables (Y)

- Train_Data_shape : (252000, 13)
- It has **252,000 samples and 11 features.**
- Independent variables are used to predict of Risk_Flag which is dependent variables.
- Risk_Flag(Y) is binary clas (0 or 1) .

Method

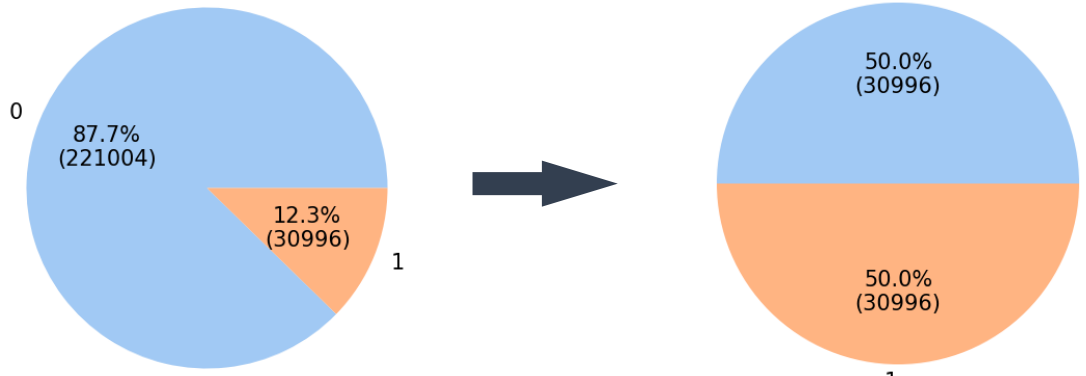
Pre-processing

1. Label Encoding → 2. Down sampling → 3. Standard Scaling

pre_processed

											CURRENT_HOUSE_YRS
0	1										-1.423349
1	-1										-0.709898
2	-1										0.003553
3	0										1.430456
4	-1										0.003553
...											...
12394	0										0.003553
12395	-0										-0.709898
12396	-1.3										-0.709898
12397	1.308391	-1.544535	0.523834	0.324361	0.259797	-0.633161	-0.817790	-0.625076	1.202801	0.745670	-1.423349
12398	0.717657	1.710594	1.186682	0.324361	0.259797	-0.633161	-0.412333	-0.919165	0.346602	1.832494	0.003553

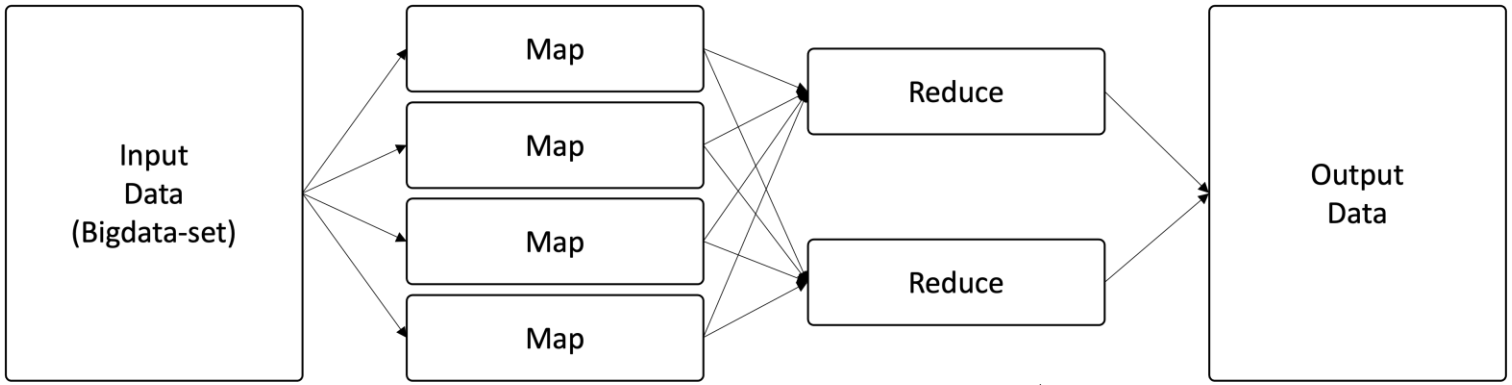
12399 rows × 11 columns



This process is required to prevent model learning bias.

Method

MapReduce functional programming




```python
def split_data_into_partitions(X, y, num_partitions):
    data_partitions = []
    chunk_size = len(X) // num_partitions

    for i in range(num_partitions):
        start_idx = i * chunk_size
        end_idx = (i + 1) * chunk_size
        X_partition = X[start_idx:end_idx]
        y_partition = y[start_idx:end_idx]
        data_partitions.append((X_partition, y_partition))

    return data_partitions
```

```python
def map_function(data_partition, params):
    X, y = data_partition
    gradients = np.dot(X.T, np.dot(X, params) - y)

    return gradients
```

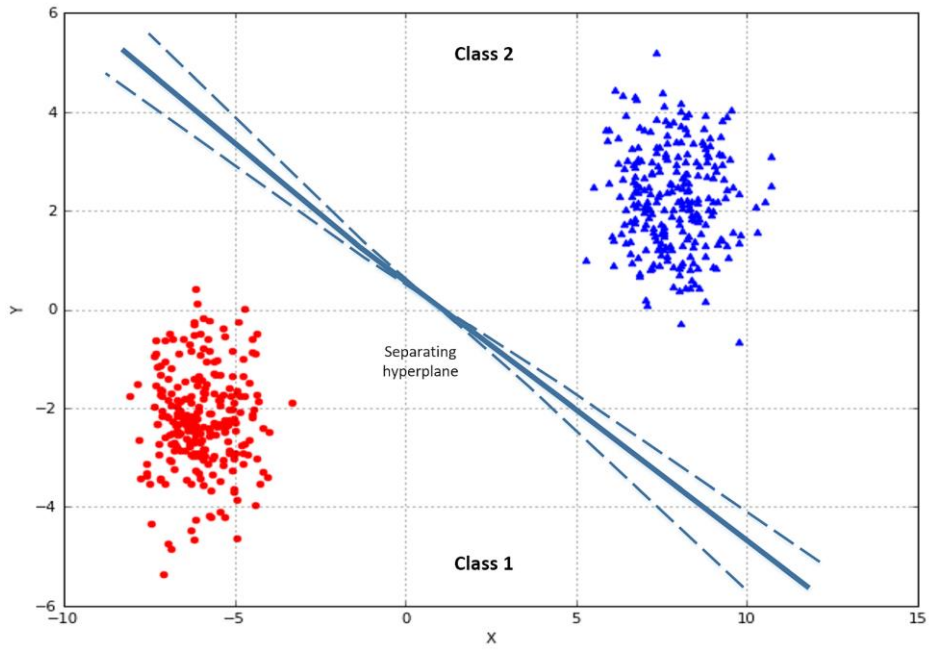
```python
def reduce_function(intermediate_results, learning_rate):
    total_gradients = np.sum(intermediate_results, axis=0)
    updated_params = learning_rate * total_gradients

    return updated_params
```

Method

Classification

➡ **Applying for Linear model !**

Examples of Linear model [4]



- **Ordinary Least Squares (OLS)**

: estimate the unknown parameters (b) in linear regression model.

$$\mathbf{y} = \mathbf{Xb} + \mathbf{e}$$

$$J = \mathbf{e}^{\mathrm{T}}\mathbf{e} = (\mathbf{y} - \mathbf{Xb})'(\mathbf{y} - \mathbf{Xb})$$
$$= (\mathbf{y}' - \mathbf{b}'\mathbf{X}')(\mathbf{y} - \mathbf{Xb})$$
$$= \mathbf{y}'\mathbf{y} - \mathbf{y}'\mathbf{Xb} - \mathbf{b}'\mathbf{X}'\mathbf{y} + \mathbf{b}'\mathbf{X}'\mathbf{Xb}$$
$$= \mathbf{y}'\mathbf{y} - 2\mathbf{b}'\mathbf{X}'\mathbf{y} + \mathbf{b}'\mathbf{X}'\mathbf{Xb}$$

$$\frac{\partial \mathbf{e}'\mathbf{e}}{\partial \mathbf{b}} = -2\mathbf{X}'\mathbf{y} + 2\mathbf{X}'\mathbf{Xb} = 0$$

$$(\mathbf{X}'\mathbf{X})\mathbf{b} = \mathbf{X}'\mathbf{y}$$

$$\mathbf{b} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

Results

Pycharm M1	JupyterLab
Pool worker: 8	Pool worker: 48
Multiprocessing Time: 4.034min 2.028sec	Multiprocessing Time: 1.065min 3.902sec

- **Jupyter** Server was able to obtain **faster results**,
  because it performs parallel processing , according to the number of cores of the CPU.

Discussion

- As the result of this paper,
  Using the Jupyter server is much faster than processing with local server,
  when we handle the big-data, which has about 250,000 samples.

=> Therefore, we are going to test the model of this paper with a lot bigger data sets.
(Use large amounts of data that are not even stored on the local server)

Reference

[1] Hiba Basim Alwan and Ku Ruhana Ku-Mahamud 2020 *IOP Conf. Ser.: Mater. Sci. Eng.* **769** 012007
[2] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. 2010. Spark: cluster computing with working sets. In Proceedings of the 2nd USENIX conference on Hot topics in cloud computing (HotCloud'10). USENIX Association, USA, 10.
[3] Data Reference : https://www.kaggle.com/datasets/subhamjain/loan-prediction-based-on-customer-behavior?resource=download&select=Sample+Prediction+Dataset.csv
[4] Image Reference : https://subscription.packtpub.com/book/data/9781785889622/5/ch05lvl1sec39/linear-classification