Documentation VLBA II - System Architectures

Scenario III: Research & Solution Development

Amro Abdalla [redacted]
Sasan Haidar Vasim 230089
Vishnu Devadas [redacted]
Magdeburg 3/7/2022

Otto-von-Guericke-Universität Magdeburg
Magdeburg Research and Competence Cluster
Fakultät für Informatik
Institut für Technische und Betriebliche Informationssysteme
Arbeitsgruppe Wirtschaftsinformatik

1. Table of contents

Scenario III: Research & Solution Development 1
1. Table of contents 2
2. Table of figures 4
3. List of Abbreviations 7
1. Introduction 1
2. Analytics using BigQuery 2
    a. 2.1 Data Preparation (Views) 3
    b. 2.2 Visualization 11
3. Using Machine Learning To Predict The Best Candidate 18
    a. 3.1 Creating BigQuery view for training 18
    b. 3.2 Creating the machine learning model 20
    c. 3.3 Trigger the Vertex AI pipeline using Pub/Sub 25
4. Future Project Trends Forecast using Historical Data and Google Trends 30
    d. 4.1 Data Preparation (BigQuery and Google Trends) 30
        i. 4.1.1 View from Historical Data using BigQuery 30
        ii. 4.1.2 Google Trends Data 31
    e. 4.2 Machine Learning Models (VertexAI) 32
        i. 4.2.1 Datasets 32
        ii. 4.2.2 Training 33
        iii. 4.2.3 Models 34
    f. 4.3 Batch Predictions (VertexAI) 34
    g. 4.4 Visualization and Data Analysis (Google Data Studio) 35
        i. 4.4.1 Forecast Data 36
        ii. 4.4.2 Historical Data 37
        iii. 4.4.3 Forecast with the Historical Data 37
    h. 4.5 Workflow Diagrams 38
        i. 4.5.1 Data Preparation 38
        ii. 4.5.2 Machine Learning Models and Batch Predictions 38
        iii. 4.5.3 Visualization 40
    i. 4.6 Challenges and Limitations 40
        i. 4.6.1 Google Trends 40
        ii. 4.6.2 Vertex AI 41
        iii. 4.6.3 Google Data Studio 41
    j. 4.7 Google Data Studio Report 42
        i. 4.7.1 MPD Department 42

ii. 4.7.2 CCR Department 43
iii. 4.7.3 RSD Department 45
iv. 4.7.4 MS Department 46
7. Results 48
8. References 49

2. Table of figures

Number	Name	Page
1	Architecture diagram	2
2	Workflow of Analytics and Visualization	2
3	Bar Chart of success rate of employees with different academic titles.	12
4	Bar Chart of success rate of employees with different education fields.	13
5	Recommended universities for recruiting candidates in CCR	14
6	Recommended universities for recruiting candidates in MPD	15
7	Recommended universities for recruiting candidates in MS	16
8	Recommended universities for recruiting candidates in RSD	17
9	Attributes used for training the machine learning model	18
10	BigQuery ML cheat sheet	21
11	Models and their mean absolute errors	23
12	Data split for best candidate prediction model	23
13	Feature importance in Vertex AI for the best candidate prediction model	24
14	Trigger VertexAI pipeline using Cloud Pub/Sub	25
15	Google Trends Interest Over Time worldwide comparison data over a custom date range	29

16	**Creating Dataset in Vertex AI for AutoML Tabular Forecasting**	**30**
17	**Importing RSD Department dataset from BigQuery to Vertex AI**	**31**
18	**Imported Dataset of each department in Vertex AI**	**31**
19	**Example of RSD Department Models with training Finished status**	**32**
20	**Example of RSD Department Batch predictions with Finished status**	**33**
21	**Blend Data configuration for Forecast Data (RSD)**	**34**
22	**Forecast Data of Trends (RSD)**	**34**
23	**Historical Data of Trends (RSD)**	**35**
24	**Blend Data configuration for Forecast and Historical Data (RSD)**	**35**
25	**Forecast Data with the Historical Data of Trends (RSD)**	**36**
26	**Data Preparation Workflow**	**36**
27	**Vertex AI Workflow**	**37**
28	**Google Data Studio Visualization Workflow**	**38**
29	**Data inconsistency in ML models after training**	**39**
30	**MPD Trends Forecast**	**40**
31	**MPD Trends Historical**	**40**
32	**MPD Trends Historical + Forecast**	**41**
33	**CCR Trends Forecast**	**41**

34	**CCR Trends Historical**	**42**
35	**CCR Trends Historical + Forecast**	**42**
36	**RSD Trends Forecast**	**43**
37	**RSD Trends Historical**	**43**
38	**RSD Trends Historical + Forecast**	**44**
39	**MS Trends Forecast**	**44**
40	**MS Trends Historical**	**45**
41	**MS Trends Historical + Forecast**	**45**

3. List of Abbreviations

ML	Machine Learning
GCP	Google Cloud Platform
RSD	Research & Solution Development
MPD	Manufacturing & Product Distribution
MS	Maintenance & Service
CCR	Consulting & Customer Retention

1. Introduction

As a team hired in the analyst division of RSD under Mobility Worldwide, we are required to aid in the decision making process of recruitment. Mainly by using analytics and visualization capabilities of Google Cloud. In addition, we are also tasked to integrate some specific tools and external data sources in this process. The RSD department has multiple areas of expertise that organize projects for the new development of products and technologies. To ensure workforce success, highly qualified personnel are required. Therefore, a data-driven recruiting strategy needs to be implemented, along with identifying future project trends in existing business areas.

To achieve this goal, our main tasks are summarized below:

1. Using ML techniques, to identify future project trends based on past data and Google Trends.
2. Analyzing the current employee pool and outcome of past projects, to implement a data-driven recruiting strategy.
3. Recommend universities based on insights gained from the data analysis to gain better access to promising talents.

To achieve these tasks, data (CSV files) from SAP was stored into BigQuery, from which different views were made using SQL workspace (BigQuery). These were used later on for creating machine learning models using VertexAI or BigQuery ML, and also in the visualization of results using Google Data Studio.

For the purpose of forecasting future project trends in the four departments (MPD, CCR, RSD and MS), we utilized data from BigQuery by creating a view of successful projects, as well as historical data gathered from Google Trends for model creation and forecasting. The machine learning models were generated in VertexAI using Auto ML, after which batch predictions were made from it. This was then successfully stored into BigQuery for visualization using Google Data Studio [1]. Reports were created for each department, with three pages that include forecast data, historical data, and their combination.

To create a prediction model for the best candidate, different machine learning approaches were compared, from which AutoML was subsequently selected. VertexAI was then used to import datasets from BigQuery, to use it for training an AutoML model. Using VertexAI, the model can also be deployed in an endpoint that could be accessed by the client, in order to do prediction. These steps in VertexAI were repeated again but this time with python code, to automate the process of using a VertexAI pipeline. The python code generates a JSON file which will be used for deploying models through the pipeline. In order to trigger the pipeline, Cloud Pub/Sub and Cloud Function services were made use of, where the cloud function subscribes to the Pub/Sub service for the pipeline. If the client makes a publish request to this service, the Cloud Function service will trigger the pipeline.

BigQuery was also used to carry out analytical work on internal and external data, with the aim of ultimately producing visualizations that would aid in the recruiting process. This would involve storing project and university data, and representing them as a dataset. SQL queries were applied to derive meaningful views, which are visualized using Data Studio. The semantic basis of the views used for this purpose were largely dependent on the business areas, educational background, employee status, and success/failure of projects. In depth methodology and outcomes are outlined in subsections 2.1 and 2.2 respectively.

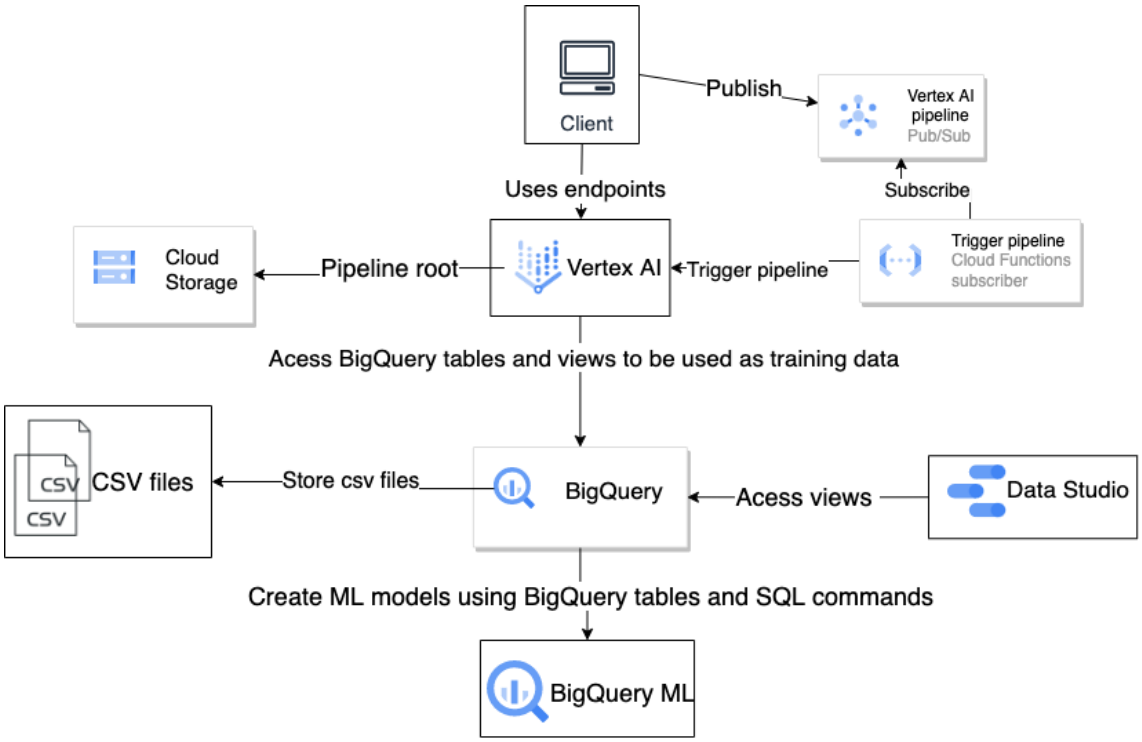


Figure 1 – Architecture diagram

2. Analytics using BigQuery

The following subsections outline the use of BigQuery analytics, and visualization of any relevant subsequent data. The goal is to answer the questions outlined in tasks 2 and 3, which investigates preferential candidates based on their qualifications, project participation, and educational background. This is carried out in two separate phases, the preparation and visualization phases respectively. A general workflow of this process is represented in the figure below :

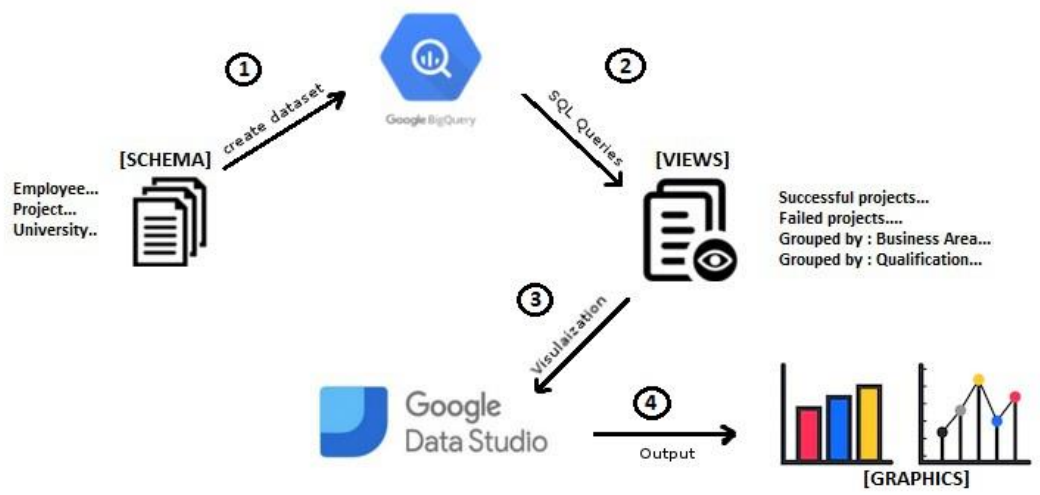


Figure 2 – Workflow of Analytics and Visualization

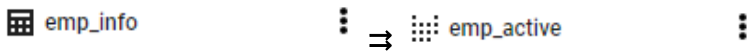
a. 2.1 Data Preparation (Views)

Before exploring the provided dataset for visualization, we required the use of semantically rich views. To get a general idea of where important information should be extracted from, we had to study every schema provided to form meaningful connections. That being said, we formulated several characteristics to be taken into account. They are as follows :

- Consider only employees that have an “active status” at the company.
- Consider only employees that hold a qualification (Bachelors, Masters, Doctoral).
- Consider only employees that graduated from an institution.
- Consider only projects that were successfully completed.

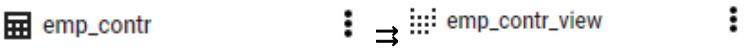
With respect to our workflow, we also took into account “failed” projects in the specified context, to get a broader understanding of all responsible characteristics. That is later addressed in step 13. For now, the following roadmap outlines our view creation for successfully completed projects, and is accompanied by its respective code-base.

1. Create a view of “active” employees.



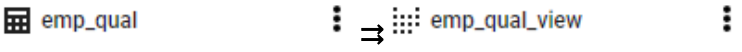

```
SELECT ei.PERNR, ei.FIRSTNA, ei.SURNA, ei.Gender
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.emp_info` ei
where ei.status='active'
order by ei.PERNR;
```

2. Create a view simplifying the employee contract schema.



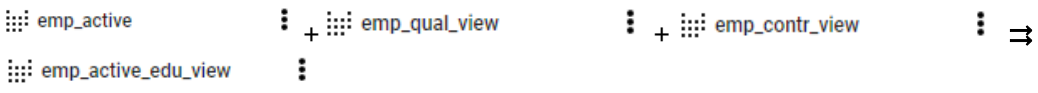

```
SELECT ec.PERNR, ec.Business_Area, ec.Payroll_Area
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.emp_contr` ec
order by ec.PERNR;
```

3. Create a view of employees with an academic degree.



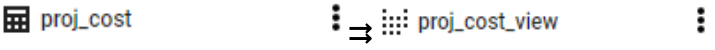

```
SELECT eq.PERNR, eq.EducationField, eq.title, eq.Institution
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.emp_qual` eq
where eq.AcademicDegree=true
order by eq.PERNR;
```

4. Join the emp_active, emp_contr_view, and emp_qual_view to create a view of "active educated" employees.



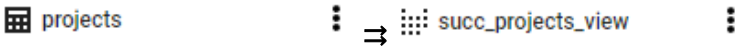

```
SELECT ea.PERNR, ea.FIRSTNA, ea.SURNA, ea.Gender,
ec.Business_Area, ec.Payroll_Area, eq.EducationField,
eq.title, eq.Institution
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.emp_active` ea,
`vlbaii-ss22-uc-3-development.VLBA_Group3.emp_contr_view` ec,
`vlbaii-ss22-uc-3-development.VLBA_Group3.emp_qual_view` eq
where ea.PERNR=ec.PERNR
and ea.PERNR=eq.PERNR
and ec.PERNR=eq.PERNR
order by ea.PERNR;
```

5. Create a view simplifying the project cost schema.



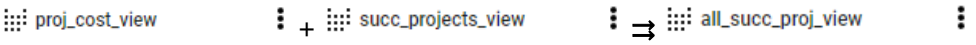

```
SELECT pc.Project_ID, pc.Currency, pc.Budget, pc.Project_Cost_real-pc.Project_Rev_real as final_cost
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.proj_cost` pc
order by pc.Project_ID;
```

6. Create a view of successful projects.



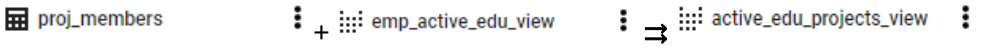

```
SELECT p.ID, p.description, p.RESNR as head_ID, p.RESNA as head_name, p.Business_Area
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.projects` p
where p.state='successful'
order by p.ID;
```

7. Join the proj_cost_view and succ_projects_view to create a view of all "successfully completed" projects.



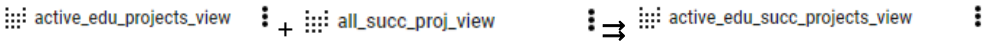

```
SELECT sp.ID, sp.description, sp.head_ID, sp.head_name, sp.Business_Area, pc.Budget, pc.final_cost, pc.Currency
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.succ_projects_view` sp, `vlbaii-ss22-uc-3-development.VLBA_Group3.proj_cost_view` pc
where sp.ID=pc.Project_ID
order by sp.ID;
```

8. Join the emp_active_edu_view and project members schema to create a view of projects participated by "active, educated" employees.



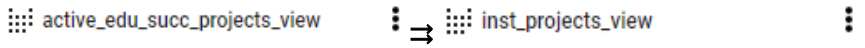

```
SELECT ae.PERNR, ae.FIRSTNA as first_name, ae.SURNA as surname,
pm.Project_ID, ae.Gender, ae.Business_Area,
ae.Payroll_Area, ae.EducationField, ae.title, ae.Institution
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.emp_active_edu_view` ae,
`vlbaii-ss22-uc-3-development.VLBA_Group3.proj_members` pm
where ae.PERNR=pm.MemberId
order by ae.PERNR;
```

9. Join the all_succ_proj_view and active_edu_projects_view, to create a view of "successful" projects with "active, educated" employees as its members.

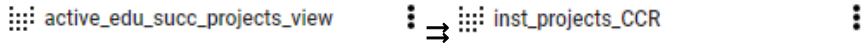



```
SELECT aep.PERNR, aep.first_name, aep.surname,
aep.Business_Area, aep.Payroll_Area,
aep.EducationField, aep.title, aep.Institution,
aep.Project_ID, sp.description, sp.head_name, sp.Budget, sp.final_cost, sp.Currency
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.active_edu_projects_view` aep,
`vlbaii-ss22-uc-3-development.VLBA_Group3.all_succ_proj_view` sp
where aep.Project_ID=sp.ID
order by aep.PERNR;
```

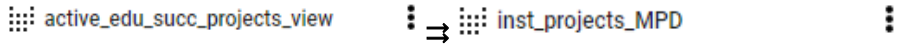
10. From the active_edu_succ_projects_view, we derive the number of successful projects by institute, and these are subcategorized by each business area (RSD, MPD, CCR, MS) respectively.



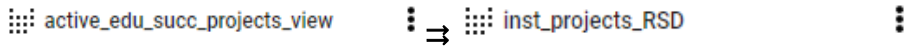

```
SELECT Institution, count(distinct Project_ID) as number_of_projects
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.active_edu_succ_projects_view`
group by Institution
order by count(distinct Project_ID) desc;
```



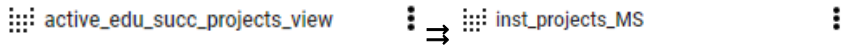

```
SELECT Institution, count(distinct Project_ID) as number_of_projects_CCR
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.active_edu_succ_projects_view` sp
where sp.Business_Area='CCR'
group by Institution
order by count(distinct Project_ID) desc;
```




```
SELECT Institution, count(distinct Project_ID) as number_of_projects_MPD
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.active_edu_succ_projects_view` sp
where sp.Business_Area='MPD'
group by Institution
order by count(distinct Project_ID) desc;
```




```
SELECT Institution, count(distinct Project_ID) as number_of_projects_RSD
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.active_edu_succ_projects_view` sp
where sp.Business_Area='RSD'
group by Institution
order by count(distinct Project_ID) desc;
```




```
SELECT Institution, count(distinct Project_ID) as number_of_projects_MS
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.active_edu_succ_projects_view` sp
where sp.Business_Area='MS'
group by Institution
order by count(distinct Project_ID) desc;
```

11. Following the active_edu_succ_projects_view from step 10, we use this view to determine successful projects based on academic qualifications (Bachelor, Master, Doctoral), and these are further decomposed based on educational fields (Economics, CS, Eng, Other) and business areas.

- Grouping by educational title (Master degree holders) :

active_edu_succ_projects_view ⇉ masters_succ_proj

```
SELECT title as edu_title, count(distinct Project_ID) as number_of_succ_projects
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.active_edu_succ_projects_view`
where title = 'Master Degree'
group by title
order by count(distinct Project_ID) desc;
```

active_edu_succ_projects_view ⇉ ba_succ_masters

```
SELECT Business_Area as business_area_masters, count(distinct Project_ID) as number_of_succ_projects
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.active_edu_succ_projects_view`
where title = 'Master Degree'
group by Business_Area
order by count(distinct Project_ID) desc;
```

active_edu_succ_projects_view ⇉ edu_field_succ_masters

```
SELECT Business_Area,EducationField as edu_field_masters, count(distinct Project_ID) as number_of_projects
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.active_edu_succ_projects_view`
where title = 'Master Degree'
group by Business_Area, EducationField
order by Business_Area;
```

- Grouping by educational title (Bachelor degree holders) :

active_edu_succ_projects_view ⇉ bachelors_succ_proj

```
SELECT title as edu_title, count(distinct Project_ID) as number_of_succ_projects
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.active_edu_succ_projects_view`
where title = 'Bachelor Degree'
group by title
order by count(distinct Project_ID) desc;
```

active_edu_succ_projects_view ⇉ ba_succ_bachelors

```
SELECT Business_Area as business_area_bachelors, count(distinct Project_ID) as number_of_succ_projects
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.active_edu_succ_projects_view`
where title = 'Bachelor Degree'
group by Business_Area
order by count(distinct Project_ID) desc;
```

active_edu_succ_projects_view ⇉ edu_field_succ_bachelors

```
SELECT Business_Area,EducationField as edu_field_bachelors, count(distinct Project_ID) as number_of_projects
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.active_edu_succ_projects_view`
where title = 'Bachelor Degree'
group by Business_Area,EducationField
order by Business_Area;
```

- Grouping by educational title (Doctoral degree holders) :

active_edu_succ_projects_view ⇉ doctoral_succ_proj

```
SELECT title as edu_title, count(distinct Project_ID) as number_of_succ_projects
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.active_edu_succ_projects_view`
where title = 'Doctoral Degree'
group by title
order by count(distinct Project_ID) desc;
```

active_edu_succ_projects_view ⇉ ba_succ_doctoral

```
SELECT Business_Area as business_area_doctoral, count(distinct Project_ID) as number_of_succ_projects
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.active_edu_succ_projects_view`
where title = 'Doctoral Degree'
group by Business_Area
order by count(distinct Project_ID) desc;
```

active_edu_succ_projects_view ⇉ edu_field_succ_doctoral

```
SELECT Business_Area,EducationField as edu_field_doctoral, count(distinct Project_ID) as number_of_projects
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.active_edu_succ_projects_view`
where title = 'Doctoral Degree'
group by Business_Area,EducationField
order by Business_Area;
```

12. Apart from using internal data, our problem space required exploration of external data as well. This was in the form of university rankings, and best faculty. Before merging these with our derived views, we restructured it in the following way :

- Create a view of university ranking for each respective year.

uni_ranks ⇉ uni_ranks_18

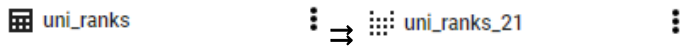
```
SELECT world_rank, institution, score
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks`
where year=2018
order by world_rank;
```

uni_ranks ⇉ uni_ranks_19

```
SELECT world_rank, institution, score
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks`
where year=2019
order by world_rank;
```

uni_ranks ⇉ uni_ranks_20

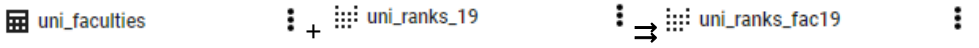
```
SELECT world_rank, institution, score
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks`
where year=2020
order by world_rank;
```



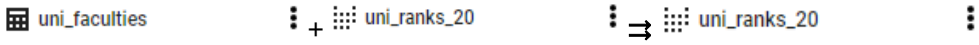

```
SELECT world_rank, institution, score
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks`
where year=2021
order by world_rank;
```

- Create views by joining each yearly view with the uni_faculty schema, to create rankings by year with its respective best faculty.


```
SELECT ur.world_rank, ur.institution, uf.faculty as best_faculty, ur.score
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks_18` ur, `vlbaii-ss22-uc-3-development.VLBA_Group3.uni_faculties_view` uf
where ur.institution=uf.institute
order by ur.world_rank;
```



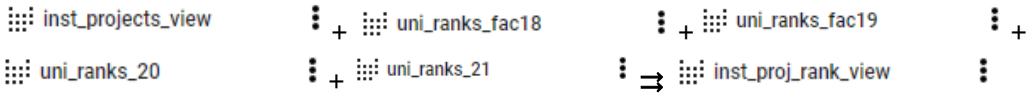

```
SELECT ur.world_rank, ur.institution, uf.faculty as best_faculty, ur.score
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks_19` ur, `vlbaii-ss22-uc-3-development.VLBA_Group3.uni_faculties_view` uf
where ur.institution=uf.institute
order by ur.world_rank;
```




```
SELECT ur.world_rank, ur.institution, uf.faculty as best_faculty, ur.score
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks_20` ur, `vlbaii-ss22-uc-3-development.VLBA_Group3.uni_faculties_view` uf
where ur.institution=uf.institute
order by ur.world_rank;
```

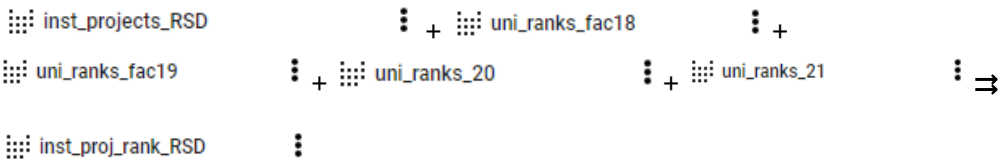

```
SELECT ur.world_rank, ur.institution, uf.faculty as best_faculty, ur.score
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks_21` ur, `vlbaii-ss22-uc-3-development.VLBA_Group3.uni_faculties_view` uf
where ur.institution=uf.institute
order by ur.world_rank;
```

- Create views of institution rank by respective years, accompanied by the total number of successful projects. In order to do so, we join the inst_projects_view with each uni_ranks_xx view. Furthermore, these are also subdivided by business area respectively.



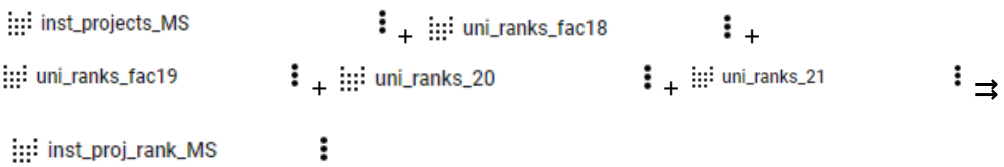

```
SELECT p.Institution, p.number_of_projects, u18.world_rank as uni_ranking18,
u19.world_rank as uni_ranking19, u20.world_rank as uni_rank20, u21.world_rank as uni_rank21
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.inst_projects_view` p,
`vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks_fac18` u18,
`vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks_fac19` u19,
`vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks_fac20` u20,
`vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks_fac21` u21
where p.Institution=u18.institution
and p.Institution=u19.institution
and p.Institution=u20.institution
and p.Institution=u21.institution
order by p.number_of_projects desc;
```

- For business area RSD :



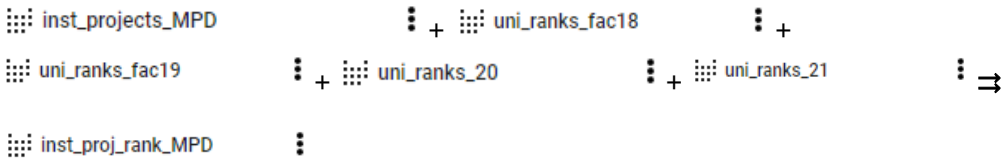

```
SELECT p.Institution, p.number_of_projects_RSD, u18.world_rank as uni_ranking18,
u19.world_rank as uni_ranking19, u20.world_rank as uni_rank20, u21.world_rank as uni_rank21
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.inst_projects_RSD` p,
`vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks_fac18` u18,
`vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks_fac19` u19,
`vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks_fac20` u20,
`vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks_fac21` u21
where p.Institution=u18.institution
and p.Institution=u19.institution
and p.Institution=u20.institution
and p.Institution=u21.institution
order by p.number_of_projects_RSD desc;
```

- For business area MS :



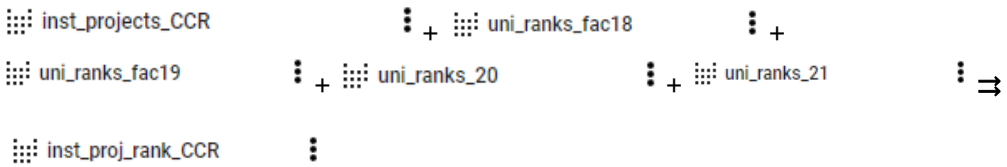

```
SELECT p.Institution, p.number_of_projects_MS, u18.world_rank as uni_ranking18,
u19.world_rank as uni_ranking19, u20.world_rank as uni_rank20, u21.world_rank as uni_rank21
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.inst_projects_MS` p,
`vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks_fac18` u18,
`vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks_fac19` u19,
`vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks_fac20` u20,
`vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks_fac21` u21
where p.Institution=u18.institution
and p.Institution=u19.institution
and p.Institution=u20.institution
and p.Institution=u21.institution
order by p.number_of_projects_MS desc;
```

- For business area MPD :




```sql
SELECT p.Institution, p.number_of_projects_MPD, u18.world_rank as uni_ranking18,
u19.world_rank as uni_ranking19, u20.world_rank as uni_rank20, u21.world_rank as uni_rank21
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.inst_projects_MPD` p,
`vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks_fac18` u18,
`vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks_fac19` u19,
`vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks_fac20` u20,
`vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks_fac21` u21
where p.Institution=u18.institution
and p.Institution=u19.institution
and p.Institution=u20.institution
and p.Institution=u21.institution
order by p.number_of_projects_MPD desc;
```

- For business area CCR :




```sql
SELECT p.Institution, p.number_of_projects_CCR, u18.world_rank as uni_ranking18,
u19.world_rank as uni_ranking19, u20.world_rank as uni_rank20, u21.world_rank as uni_rank21
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.inst_projects_CCR` p,
`vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks_fac18` u18,
`vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks_fac19` u19,
`vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks_fac20` u20,
`vlbaii-ss22-uc-3-development.VLBA_Group3.uni_ranks_fac21` u21
where p.Institution=u18.institution
and p.Institution=u19.institution
and p.Institution=u20.institution
and p.Institution=u21.institution
order by p.number_of_projects_CCR desc;
```

13. The logic used for creating successfully completed project views is mirrored for failed projects. To avoid redundancy and repetition with minor discrepancies, the following table includes successful views and its failed project counterparts for future reference.

Successful project views / Failed project views
succ_projects_view | fail_projects_view
all_succ_proj_view | all_fail_proj_view
active_edu_succ_projects_view | active_edu_fail_projects_view
inst_projects_view | inst_fail_projects_view
inst_projects_RSD | inst_fail_projects_RSD
inst_projects_MPD | inst_fail_projects_MPD
inst_projects_CCR | inst_fail_projects_CCR
inst_projects_MS | inst_fail_projects_MS
inst_proj_rank_view | inst_fail_proj_rank_view
inst_proj_rank_RSD | inst_fail_proj_rank_RSD
inst_proj_rank_MPD | inst_fail_proj_rank_MPD
inst_proj_rank_CCR | inst_fail_proj_rank_CCR
inst_proj_rank_MS | inst_fail_proj_rank_MS
masters_succ_proj | master_fail_proj
edu_field_succ_masters | edu_field_fail_masters
ba_succ_masters | ba_fail_masters
bachelors_succ_proj | bachelor_fail_proj
edu_field_succ_bachelors | edu_field_fail_bachelors
ba_succ_bachelors | ba_fail_bachelors
doctoral_succ_proj | doctoral_fail_proj
edu_field_succ_doctoral | edu_field_fail_doctoral
ba_succ_doctoral | ba_fail_doctoral

b. 2.2 Visualization

To get visual insights from data for implementing a data-driven recruiting strategy, we made use of Data Studio for visualizing results obtained from the above SQL views.

First, we'll analyze the current employee pool and results of past projects, to know which type of employees perform better under which departments, such that we can consider these attributes while recruiting in the future.

For this reason we used the same logic as mentioned in the Machine learning section, which uses a success factor considering both the successful and failed projects by active employees. Let's first consider the success rate of employees with different academic degrees of different departments. For this, we've used the views 'ba_succ_bachelors' and 'ba_fail_bachelors' from the data preparation part.

To get our success rate, we use the following formula :

$$\frac{number\ of\ successful\ projects}{number\ of\ successful\ projects\ +\ number\ of\ failed\ projects}$$

4. We created new views for each academic title i.e. bachelors, masters and doctoral as :

ba_succ_bachelors + ba_fail_bachelors ⇉ ba_succ_bachelors_perc

```
SELECT sp.business_area_bachelors,
sp.number_of_succ_projects/(sp.number_of_succ_projects+fp.number_of_fail_projects) as success_percentage
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.ba_succ_bachelors` sp,
`vlbaii-ss22-uc-3-development.VLBA_Group3.ba_fail_bachelors` fp
where sp.business_area_bachelors=fp.business_area_bachelors
```

ba_succ_doctoral + ba_fail_doctoral ⇉ ba_succ_doctoral_perc

```
SELECT sp.business_area_doctoral,
sp.number_of_succ_projects/(sp.number_of_succ_projects+fp.number_of_fail_projects) as success_percentage
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.ba_succ_doctoral` sp,
`vlbaii-ss22-uc-3-development.VLBA_Group3.ba_fail_doctoral` fp
where sp.business_area_doctoral=fp.business_area_doctoral
```

ba_succ_masters + ba_fail_masters ⇉ ba_succ_masters_perc

```
SELECT sp.business_area_masters,
sp.number_of_succ_projects/(sp.number_of_succ_projects+fp.number_of_fail_projects) as success_percentage
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.ba_succ_masters` sp,
`vlbaii-ss22-uc-3-development.VLBA_Group3.ba_fail_masters` fp
where sp.business_area_masters=fp.business_area_masters
```

Each of these views was plotted on a bar chart as displayed below :

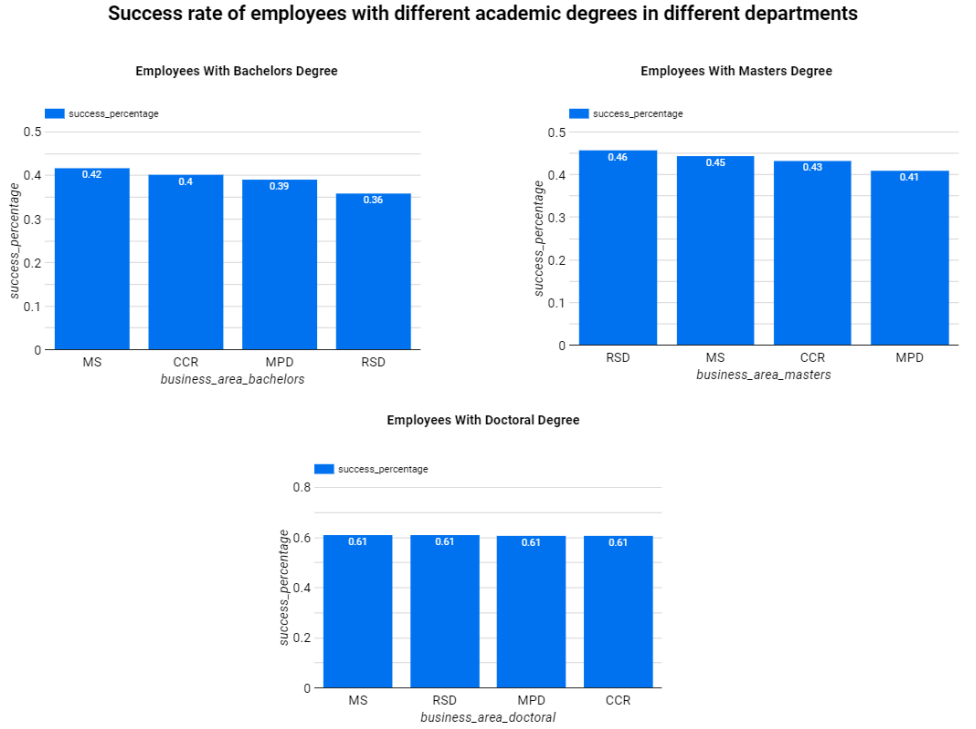


Figure 3 – Bar Chart of success rate of employees with different academic titles.

5. After getting an insight from academic degrees, we proceeded to visualize the educational fields of employees in all departments. For that, we needed to create 3 new views in order to get a success rate, using the previously created views of data preparation as illustrated :

edu_field_succ_bachelors + edu_field_fail_bachelors ⇉ edu_field_bachelors_perc

```
select sp.Business_Area, sp.edu_field_bachelors, sp.number_of_projects/(sp.number_of_projects+fp.number_of_fail_projects) success_perc
from `vlbaii-ss22-uc-3-development.VLBA_Group3.edu_field_succ_bachelors` sp,`vlbaii-ss22-uc-3-development.VLBA_Group3.edu_field_fail_bachelors` fp
where sp.Business_Area=fp.Business_Area and
sp.edu_field_bachelors=fp.edu_field_bachelors
order by sp.Business_Area
```

edu_field_fail_masters + edu_field_succ_masters ⇉ edu_field_masters_perc

```
select sp.Business_Area, sp.edu_field_masters, sp.number_of_projects/(sp.number_of_projects+fp.number_of_fail_projects) success_perc
from `vlbaii-ss22-uc-3-development.VLBA_Group3.edu_field_succ_masters` sp,`vlbaii-ss22-uc-3-development.VLBA_Group3.edu_field_fail_masters` fp
where sp.Business_Area=fp.Business_Area and
sp.edu_field_masters=fp.edu_field_masters
order by sp.Business_Area
```

edu_field_succ_doctoral + edu_field_fail_doctoral ⇉ edu_field_doctoral_perc

```
select sp.Business_Area, sp.edu_field_doctoral, sp.number_of_projects/(sp.number_of_projects+fp.number_of_fail_projects) success_perc
from `vlbaii-ss22-uc-3-development.VLBA_Group3.edu_field_succ_doctoral` sp,`vlbaii-ss22-uc-3-development.VLBA_Group3.edu_field_fail_doctoral` fp
where sp.Business_Area=fp.Business_Area and
sp.edu_field_doctoral=fp.edu_field_doctoral
order by sp.Business_Area
```

Figure 4 – Bar Chart of success rate of employees with different education fields.

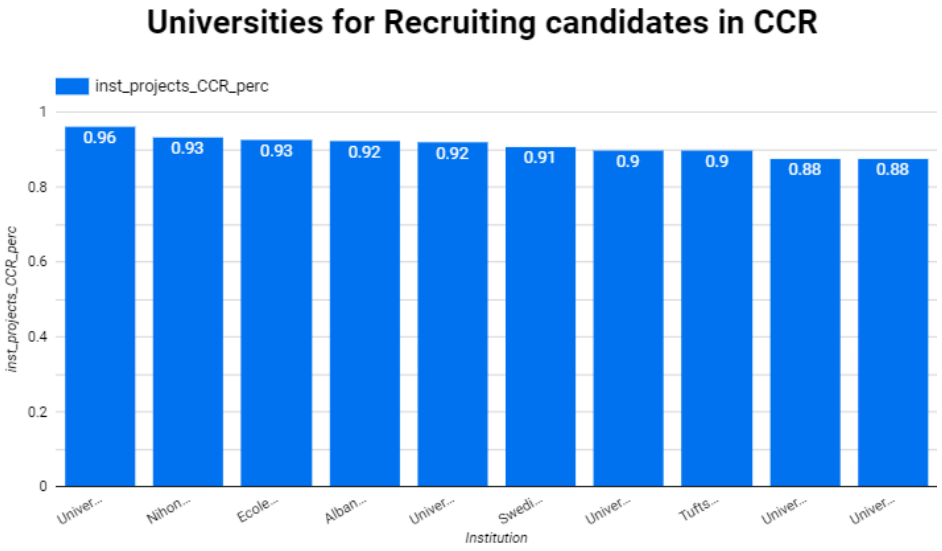
From the above visuals, we can easily interpret the success rate of employees from various educational backgrounds of respective business areas. Thus, we can give preference to candidates from high success rate fields when recruiting for respective business areas.

6. As mentioned in task three, we need to get insights from the university data as well. This is important to determine which institutions are advantageous when recruiting candidates for different business units. Similarly, we utilized the success factor from the results of successful and failed projects, and distributed it according to institution. We created additional views on top of previously made views in our data prep, as illustrated :

i. For CCR business area

inst_projects_CCR + inst_fail_projects_CCR ⇉ inst_projects_CCR_perc

```
SELECT sp.Institution,fac.faculty as best_faculty,
sp.number_of_projects_CCR/(sp.number_of_projects_CCR+fp.number_of_failed_projects_CCR) as inst_projects_CCR_perc,
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.inst_projects_CCR` sp,
`vlbaii-ss22-uc-3-development.VLBA_Group3.inst_fail_projects_CCR` fp,
`vlbaii-ss22-uc-3-development.VLBA_Group3.uni_faculties_view` fac
where sp.Institution=fp.Institution
and sp.Institution=fac.institute
order by sp.number_of_projects_CCR/(sp.number_of_projects_CCR+fp.number_of_failed_projects_CCR) desc;
```



	Institution	best_faculty	inst_projects_CCR_perc
1.	University of Virginia	Economics	0.96
2.	Nihon University	Economics	0.93
3.	Ecole Polytechnique	Economics	0.93
4.	Albany Medical College	Engineering	0.92
5.	University of Utah	Other	0.92
6.	Swedish University of Agricultural Sciences	Economics	0.91
7.	Tufts University	Computer Science	0.9
8.	University of Lausanne	Engineering	0.9
9.	University of California, Davis	Computer Science	0.88
10.	Vilnius University	Engineering	0.88
11.	University of California, Berkeley	Engineering	0.88
12.	Wayne State University	Engineering	0.88

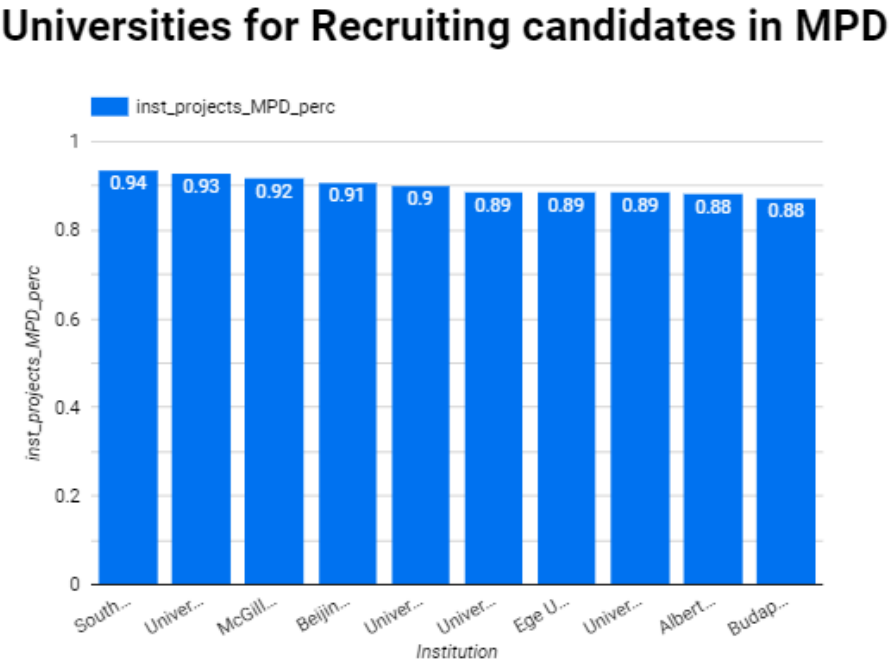
1 - 100 / 706 < >

Figure 5 – Recommended universities for recruiting candidates in CCR

ii. For MPD business area:

inst_projects_MPD + inst_fail_projects_MPD ⇉ inst_projects_MPD_perc

```
SELECT sp.Institution,fac.faculty as best_faculty,
sp.number_of_projects_MPD/(sp.number_of_projects_MPD+fp.number_of_failed_projects_MPD) as inst_projects_MPD_perc,
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.inst_projects_MPD` sp,
`vlbaii-ss22-uc-3-development.VLBA_Group3.inst_fail_projects_MPD` fp,
`vlbaii-ss22-uc-3-development.VLBA_Group3.uni_faculties_view` fac
where sp.Institution=fp.Institution
and sp.Institution=fac.institute
order by sp.number_of_projects_MPD/(sp.number_of_projects_MPD+fp.number_of_failed_projects_MPD) desc;
```



	Institution	best_faculty	inst_proj...
1.	Southeast University	Other	0.94
2.	University of Melbourne	Computer Scien...	0.93
3.	McGill University	Economics	0.92
4.	Beijing Normal University	Computer Scien...	0.91
5.	University of Missouri-Columbia	Engineering	0.9
6.	University of Strasbourg	Computer Scien...	0.89
7.	Ege University	Economics	0.89
8.	University of Texas at Austin	Economics	0.89
9.	Albert Ludwig University of Freiburg	Other	0.88
10.	University of Perpignan Via Domitia	Engineering	0.88

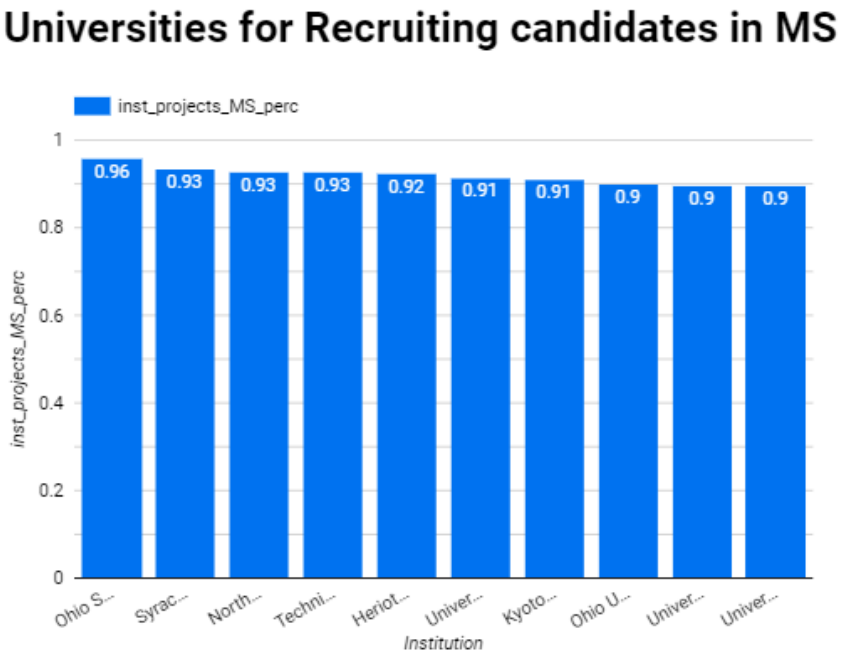
1 - 100 / 741 < >

Figure 6 – Recommended universities for recruiting candidates in MPD

ii. For MS business area:

inst_projects_MS + inst_fail_projects_MS ⇉ inst_projects_MS_perc

```
1  SELECT sp.Institution,fac.faculty as best_faculty,
2  sp.number_of_projects_MS/(sp.number_of_projects_MS+fp.number_of_failed_projects_MS) as inst_projects_MS_perc,
3  FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.inst_projects_MS` sp,
4  `vlbaii-ss22-uc-3-development.VLBA_Group3.inst_fail_projects_MS` fp,
5  `vlbaii-ss22-uc-3-development.VLBA_Group3.uni_faculties_view` fac
6  where sp.Institution=fp.Institution
7  and sp.Institution=fac.institute
8  order by sp.number_of_projects_MS/(sp.number_of_projects_MS+fp.number_of_failed_projects_MS) desc;
```



	Institution	best_faculty	inst_proje...
1.	Ohio State University, Columbus	Engineering	0.96
2.	Syracuse University	Computer Scien...	0.93
3.	North Carolina State University	Engineering	0.93
4.	Technion – Israel Institute of Technology	Engineering	0.93
5.	Heriot-Watt University	Other	0.92
6.	University College London	Other	0.91
7.	Kyoto University	Other	0.91
8.	Ohio University	Computer Scien...	0.9
9.	University at Albany, SUNY	Computer Scien...	0.9
10.	University of Georgia	Engineering	0.9
11.	Sharif University of Technology	Computer Scien...	0.89

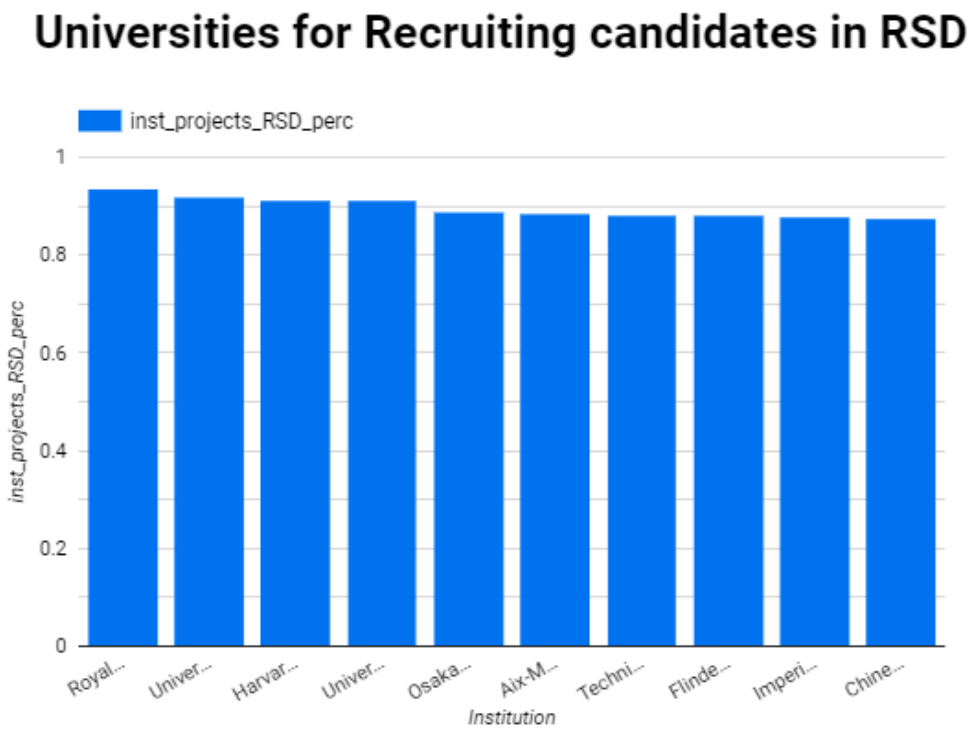
1 - 100 / 722 < >

Figure 7 – Recommended universities for recruiting candidates in MS

ii. For RSD business area:

inst_projects_RSD + inst_fail_projects_RSD ⇉ inst_projects_RSD_perc

```
SELECT sp.Institution,fac.faculty as best_faculty,
sp.number_of_projects_RSD/(sp.number_of_projects_RSD+fp.number_of_failed_projects_RSD) as inst_projects_RSD_perc,
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.inst_projects_RSD` sp,
`vlbaii-ss22-uc-3-development.VLBA_Group3.inst_fail_projects_RSD` fp,
`vlbaii-ss22-uc-3-development.VLBA_Group3.uni_faculties_view` fac
where sp.Institution=fp.Institution
and sp.Institution=fac.institute
order by sp.number_of_projects_RSD/(sp.number_of_projects_RSD+fp.number_of_failed_projects_RSD) desc;
```



	Institution	best_facu...	inst_projects...
4.	University of Cambridge	Economics	0.91
5.	Osaka University	Engineering	0.89
6.	Aix-Marseille University	Computer S...	0.88
7.	Flinders University	Engineering	0.88
8.	Technical University of Berlin	Engineering	0.88
9.	Imperial College London	Other	0.88
10.	Chinese University of Hong Kong	Engineering	0.88
11.	Cornell University	Engineering	0.87
12.	University of Dusseldorf	Other	0.87
13.	Swedish University of Agricultural Sciences	Economics	0.87

1 - 100 / 749 < >

Figure 8 – Recommended universities for recruiting candidates in RSD

From the above 4 views, we can determine for each business area, which universities have a record of highly successful candidates. In addition, the best faculty of an institution is also represented. Thus, preference of recruitment is given to candidates of these universities, while knowing their best faculty additionally helps expose what educational background these candidates originate from.

With this, we've successfully provided a means of reference for the recruiters, by gathering required insights, and representing them visually via analytics and visualization.

3. Using Machine Learning To Predict The Best Candidate

As mentioned in the previous chapter, the employee success rate was used to calculate each current employee's success, in this chapter, a view will be made that contains each employee's information with its success rate. Then different BigQuery ML approaches were used to create machine learning models to find the optimal solution. Found that AutoML gives the lowest error, so in VertexAI the created view was imported and the AutoML model was made with an endpoint to use it. Finally, a pipeline was made that can be triggered using Cloud Function, by subscribing to Cloud Pub/Sub service.

a. 3.1 Creating BigQuery view for training

In order to develop a data-driven recruiting strategy, first of all, the employee pool was analyzed in order to find which attributes can be used as a machine learning model parameter to predict suitable candidates. Data was collected from three BigQuery tables, namely emp_info, emp_contr, and emp_qual. These attributes are shown in the figure below:

Field name	Type	Mode
PERNR	INTEGER	NULLABLE
Gender	STRING	NULLABLE
Birth_date	DATE	NULLABLE
Country_of_Birth	STRING	NULLABLE
Nationality	STRING	NULLABLE
Marital_Status_Key	STRING	NULLABLE
Business_Area	STRING	NULLABLE
Payroll_Area	STRING	NULLABLE
Position	STRING	NULLABLE
EducationField	STRING	NULLABLE

Figure 9 - Attributes used for training the machine learning model

In order to save these attributes together in a new view (emp_eval_info) in BigQuery the following SQL command was executed:

```
SELECT  emp_inf.PERNR, emp_inf.Gender, emp_inf.Birth_date, emp_inf.Country_of_Birth,
emp_inf.Nationality, emp_inf.Marital_Status_Key,

emp_contr.Business_Area, emp_contr.Payroll_Area, emp_contr.Position,

emp_qual.EducationField, emp_qual.Institution, emp_qual.title

FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.emp_info` emp_inf, `vlbaii-ss22-uc-3-
development.VLBA_Group3.emp_contr` emp_contr, `vlbaii-ss22-uc-3-
development.VLBA_Group3.emp_qual` emp_qual

where emp_inf.PERNR = emp_contr.PERNR and emp_inf.PERNR=emp_qual.PERNR;
```

Then in order to predict the best candidate, a success factor is made in order to identify how successful a future candidate could be. This success factor is a value between 0 and 1, where an increase in value means increase in successfulness. It is calculated as :$\frac{number\ of\ successful\ projects}{number\ of\ successful\ projects + number\ of\ failed\ projects}$.

So in order to calculate it, two views were made. The first one was to calculate the number of failed projects for each employee, and the latter was to calculate the number of successful projects.

For the number of failed projects the view emp_num_failed_projs was made using the following query :

```
SELECT proj_mem.MemberId, count(proj.ID) as number_failed_projs FROM `vlbaii-ss22-uc-3-
development.trends.projects` proj,
`vlbaii-ss22-uc-3-development.VLBA_Group3.proj_members` proj_mem
where proj.ID =proj_mem.Project_ID and proj.state="failed"
group by proj_mem.MemberId
order by number_failed_projs;
```

And for the number of successful projects, the view emp_num_succ_projs was made using the following query:

```
SELECT proj_mem.MemberId, count(proj.ID) as number_succ_projs FROM `vlbaii-ss22-uc-3-
development.trends.projects` proj,
`vlbaii-ss22-uc-3-development.VLBA_Group3.proj_members` proj_mem
where proj.ID =proj_mem.Project_ID and proj.state="successful"
group by proj_mem.MemberId
order by number_succ_projs;
```

Then using those two views, the third view(emp_succ_score) was made to made the success factor as mentioned, using the following query:

```
select emp_succ_projs.MemberId, ROUND(emp_succ_projs.number_success_projects/
(emp_succ_projs.number_success_projects + emp_failed_projs.number_failed_projs), 2) as succ_score from
`vlbaii-ss22-uc-3-development.VLBA_Group3.emp_num_failed_projs` emp_failed_projs, `vlbaii-ss22-uc-3-
development.VLBA_Group3.emp_number_succ_projs` emp_succ_projs
where emp_failed_projs.MemberId = emp_succ_projs.MemberId
order by  succ_score;
```

After that the two views emp_eval_info and emp_succ_score were combined in order to form the view(emp_eval_info_with_score), that will be used later for making the machine learning model, as shown in the SQL command below:

```
select * from `vlbaii-ss22-uc-3-development.VLBA_Group3.emp_succ_score` emp_succ_score, `vlbaii-ss22-uc-3-
development.VLBA_Group3.emp_eval_info` eval_info
where eval_info.PERNR = emp_succ_score.MemberId
order by eval_info.PERNR;
```

b. 3.2 Creating the machine learning model

In order to build the machine learning model, BigQuery ML (Machine Learning), which is a way to create and execute machine learning models in BigQuery using SQL commands [2]. According to GCP documentation, when predicting values, they recommend using one of four options as shown in figure 2 below, namely Linear Regression, Boosted Trees Regressor, AutoML Table Regressor and Wide & Deep Regressor.

First of all, the DNN(Deep Neural Network) was used to build the ML model as shown in the SQL command below, and the mean absolute error was **1.69**.

```
CREATE OR REPLACE MODEL `vlbaii-ss22-uc-3-development.VLBA_Group3.DNN_emp_score_prejection`
OPTIONS(MODEL_TYPE='DNN_REGRESSOR',
    ACTIVATION_FN = 'RELU',
    BATCH_SIZE = 60,
    DROPOUT = 0.1,
    EARLY_STOP = TRUE,
    HIDDEN_UNITS = [68, 8],
    INPUT_LABEL_COLS = ['succ_score'],
    LEARN_RATE=0.1,
    MAX_ITERATIONS = 5,
    OPTIMIZER = 'ADAGRAD')
AS SELECT * except (PERNR, MemberId) FROM `vlbaii-ss22-uc-3-
development.VLBA_Group3.emp_eval_info_with_score`;
```

Regarding linear regression, first of all, all attributes were used to predict the success factor(succ_score)

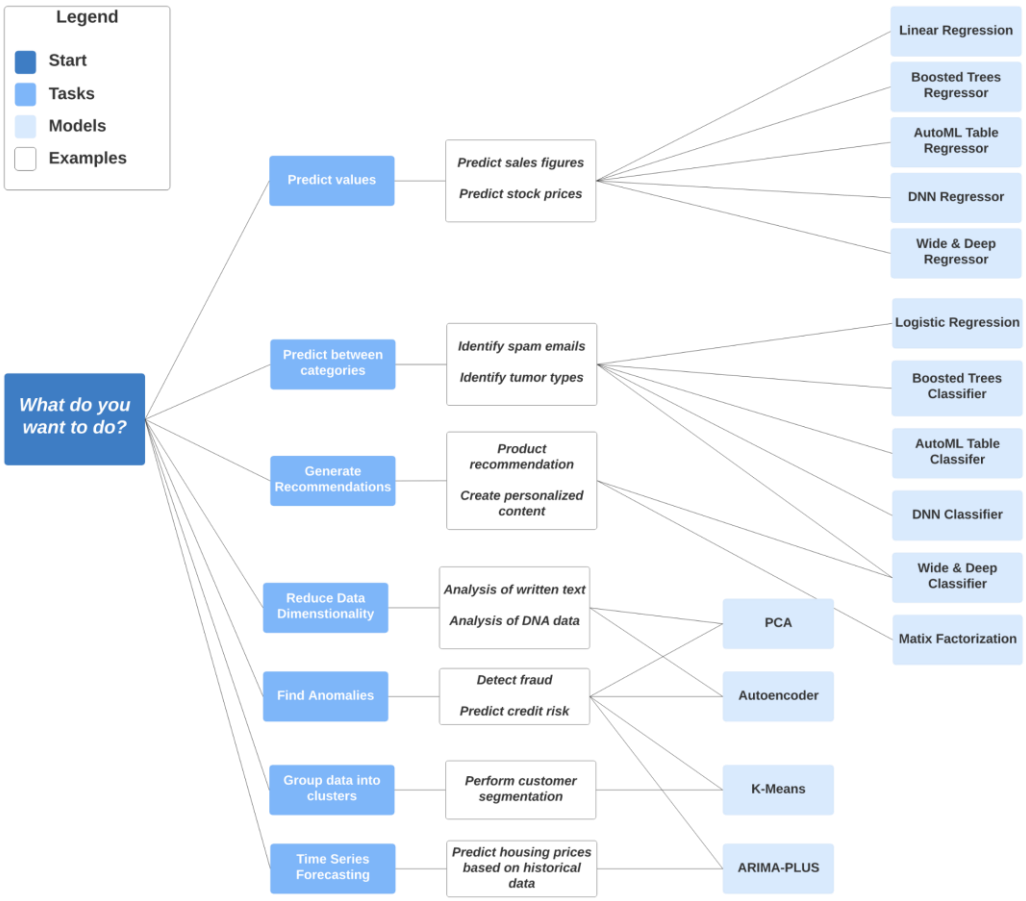


Figure 10 - BigQuery ML cheat sheet [2]

```
CREATE OR REPLACE MODEL `vlbaii-ss22-uc-3-development.VLBA_Group3.emp_score_prejection_with_all_data`
OPTIONS(
    MODEL_TYPE='LINEAR_REG',
    INPUT_LABEL_COLS = ['succ_score'],
    LS_INIT_LEARN_RATE=0.1,
    DATA_SPLIT_EVAL_FRACTION=0.05,
    DATA_SPLIT_METHOD="RANDOM",
    MAX_ITERATIONS = 5)
AS SELECT * except (PERNR, MemberId) FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.emp_eval_info_with_score`;
```

After training the mean absolute error was **0.143**, and after executing some attributes like Gender, Birth_date, Nationality, Country_of_Birth, Payroll_Area, the error was reduced to **0.140**. The SQL command for the model can be seen below:

```
CREATE OR REPLACE MODEL `vlbaii-ss22-uc-3-development.VLBA_Group3.emp_score_prejection`
OPTIONS(
    MODEL_TYPE='LINEAR_REG',
    INPUT_LABEL_COLS = ['succ_score'],
    LS_INIT_LEARN_RATE=0.1,
    DATA_SPLIT_EVAL_FRACTION=0.05,
    DATA_SPLIT_METHOD="RANDOM",
    MAX_ITERATIONS = 5)
AS SELECT * except (PERNR, Gender, Birth_date, Nationality, Country_of_Birth, MemberId, Payroll_Area)
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.emp_eval_info_with_score`;
```

After that AUTOML was tried, using the following SQL command, AUTOML got the lowest mean error **0.1272**

```
CREATE OR REPLACE MODEL `vlbaii-ss22-uc-3-
development.VLBA_Group3.AUTO_ML__candidate_emp_score_prediction`
OPTIONS(model_type='AUTOML_REGRESSOR',
        input_label_cols=['succ_score'],
        budget_hours=1.0)
AS SELECT *  except (PERNR, Gender, Birth_date, Nationality,MemberId)  FROM `vlbaii-ss22-uc-3-
development.VLBA_Group3.emp_eval_info_with_score`;
```

So to summarize the results, in figure 3 below each model and its mean absolute error.

Model	Mean absolute error
DNN	1.69
Regression	0.1424
Regression excluding (Birth, Gender, Nationality, Country of birth, Payroll_Area)	0.1362
AUTOML	0.1272

Figure 11 - Models and their mean absolute errors

Then the AUTOML model was saved in Cloud Storage, and the same process was repeated in Vertex AI, which is an environment provided by GCP for training, experiments, deploying and monitoring ML models [3]. After moving the BigQuery view there, and before the process of training there, 80% of the data was used for training, 10% for validation, and 10% for testing, and shown in the figure 4 below, the feature weighted column was not used as the value for these columns have to be numeric values.

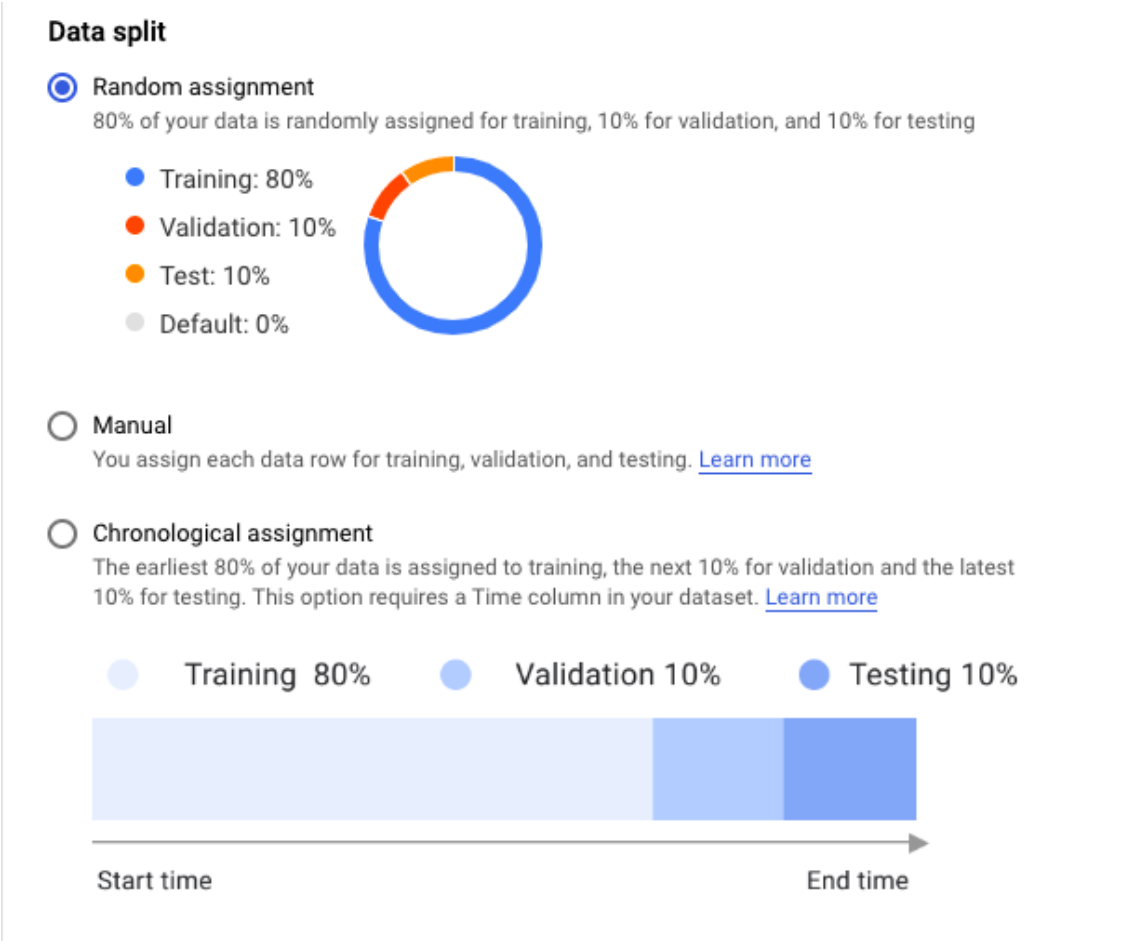


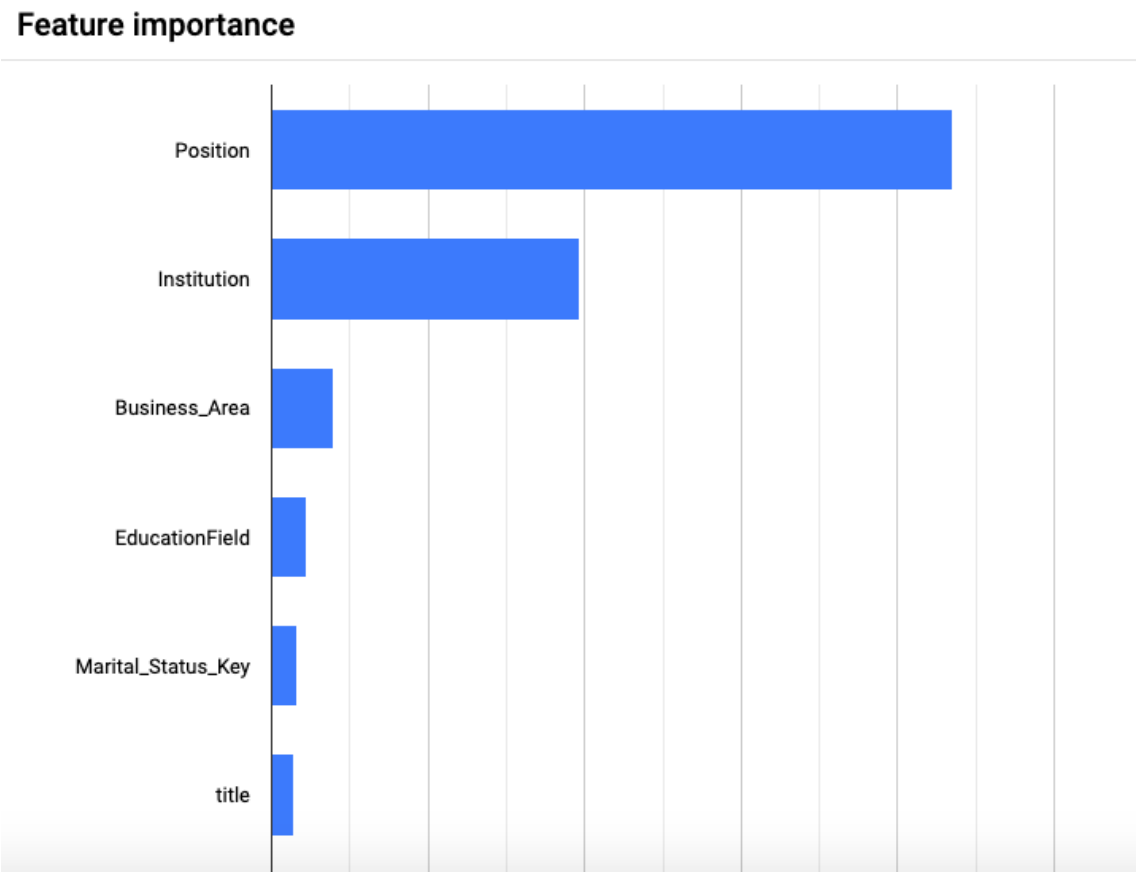
Figure 12 - Data split for best candidate prediction model

After 2 hours of training, the model gave **0.126** mean absolute error, and in Vertex AI a graph can be seen for the most important attributes in training, as can be seen in figure 5 below, position and institute were the most important.

Figure 13 - Feature importance in Vertex AI for the best candidate prediction model

Finally the model was deployed as an endpoint using the Endpoints service in Vertex AI and can be used for prediction using any scripting language or CURL by doing the following command:

```
curl \
-X POST \
-H "Authorization: Bearer $(gcloud auth print-access-token)" \
-H "Content-Type: application/json" \
https://us-central1-aiplatform.googleapis.com/v1/projects/${PROJECT_ID}/locations/us-central1/endpoints/${ENDPOINT_ID}:predict \
-d "@${INPUT_DATA_FILE}"
```



c. 3.3 Trigger the Vertex AI pipeline using Pub/Sub

As we have already created our Machine Learning model, now we have to automate the process. We can do that by creating a Machine Learning pipeline. A Machine Learning pipeline allows input to be transformed or correlated into a model that can be examined to produce output, automating the machine learning workflow. It can also be used to separate machine learning operations into separate, reusable, modular components that can be combined to create a model. This process falls under the domain of machine learning operations (MLOps).

MLOps is the practice of applying DevOps strategies to machine learning systems. Using DevOps techniques we can quickly create and distribute code updates as well as monitor systems to ensure that the reliability goals are being met. By extending this technique, machine learning operations enable us to more quickly and reliably deploy models in production while still enabling us to track and comprehend our machine learning system.

Vertex AI Pipelines help to automate, monitor and govern ML systems by orchestrating ML workflow in a serverless manner and storing workflow's artifacts using Vertex ML Metadata.

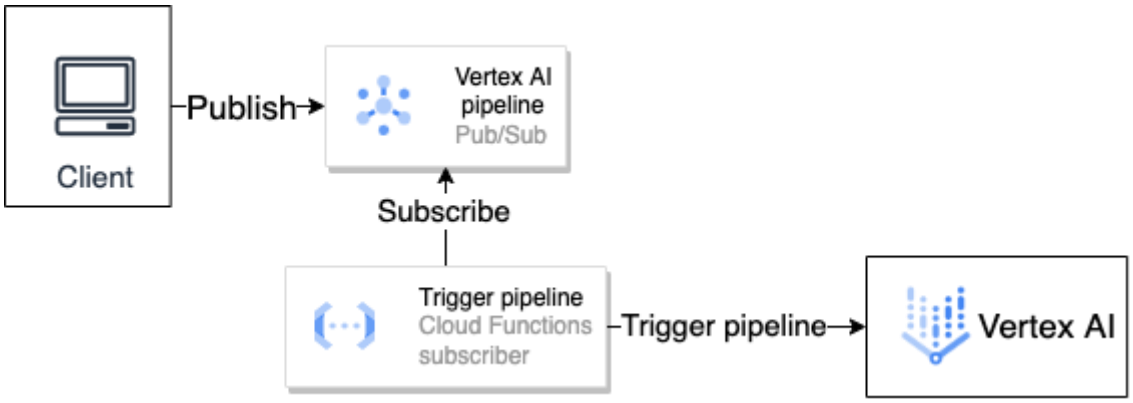


Figure 14 - Trigger VertexAI pipeline using Cloud Pub/Sub

Before we use Vertex AI pipelines to automate our machine learning pipelines, we must set up our Google Cloud Project accordingly. First, we need to configure a service account and grant access to VertexAI because our pipeline acts with the permission of our service account. Next, we need to configure a cloud storage bucket for our pipeline artifacts because our VertexAI pipelines store the artifacts of our pipeline using cloud storage. After these steps now we can build our Vertex AI pipeline.

In order to orchestrate our ML workflow on Vertex AI pipelines first, we have to describe a workflow as a pipeline. We can define our workflow by using the Kubeflow pipelines DSL package. The 'kfp.dsl' package contains the domain-specific language (DSL) that can be used to define and interact with pipelines and components.

```
$ pip3 install --upgrade google-cloud-aiplatform  {USER_FLAG} -q
$ pip3 install -U google-cloud-storage {USER_FLAG} -q
$ pip3 install {USER_FLAG} kfp google-cloud-pipeline-components --upgrade -q
```

Kubeflow pipeline components are factory functions that create pipeline steps. Each component provides information about its implementation, inputs, and outputs.

```
from google.cloud import aiplatform
import kfp
from google.cloud import aiplatform
from google_cloud_pipeline_components import aiplatform as gcc_aip

from google_cloud_pipeline_components.v1.bigquery import (BigqueryQueryJobOp)

project_id = "vlbaii-ss22-uc-3-development"
pipeline_root_path = "gs://vertexai_pipeline_vlba"



# [START aiplatform_sdk_create_and_import_dataset_tabular_bigquery_sample]
def create_and_import_dataset_tabular_bigquery_sample(
  display_name: str,
  project: str,
  bigquery_source: str,
):

  ds_op = gcc_aip.TabularDatasetCreateOp(
      display_name=display_name,
      bq_source=bigquery_source,
      project=project,
  )
  print(ds_op.outputs['dataset'])

  return ds_op

# Define the workflow of the pipeline.
@kfp.dsl.pipeline(
  name="automl-image-training-v2",
  pipeline_root=pipeline_root_path)
def pipeline(project_id: str):
  # The first step of your workflow is a dataset generator.
  ds_op = create_and_import_dataset_tabular_bigquery_sample("test", "vlbaii-ss22-uc-3-development",
"bq://vlbaii-ss22-uc-3-development.VLBA_Group3.emp_eval_info_with_score_table")
  # The second step is a model training component. It takes the dataset
  # outputted from the first step, supplies it as an input argument to the
  # component
  training_job_run_op = gcc_aip.AutoMLTabularTrainingJobRunOp(
      project=project_id,
      display_name="train_best_candiadte_OPS",
      optimization_prediction_type="regression",
```

```
    dataset=ds_op.outputs["dataset"],
    model_display_name="best_candiadte_model_OPS",
    target_column="succ_score",
    budget_milli_node_hours=8000,
)

# The third and fourth step are for deploying the model.
create_endpoint_op = gcc_aip.EndpointCreateOp(
    project=project_id,
    display_name = "best_candiadte-endpoint_OPS",
)

model_deploy_op = gcc_aip.ModelDeployOp(
    model=training_job_run_op.outputs["model"],
    endpoint=create_endpoint_op.outputs['endpoint'],
    dedicated_resources_machine_type="n1-standard-16",
    dedicated_resources_min_replica_count=1,
    dedicated_resources_max_replica_count=1,

)
```

Once the pipeline's workflow has been established, we may proceed to compile the pipeline into a JSON format. The JSON file will contain all the details needed to run a pipeline on Vertex AI Pipelines.

```
from kfp.v2 import compiler
compiler.Compiler().compile(pipeline_func=pipeline,
    package_path='best_candidate_pipline.json')
```

The Vertex AI Python client can be used to submit and run the pipeline once the workflow has been compiled into the JSON format.

```
import google.cloud.aiplatform as aip


job = aip.PipelineJob(
    display_name="automl-image-training-v2",
    template_path="image_classif_pipeline.json",
    pipeline_root=pipeline_root_path,
    parameter_values={
        'project_id': project_id
    }
```

```
)


job.submit()
```

Afterwards, we'll create a VertexAI pipeline and specify a JSON file in the pipeline.

Now, in order to trigger our Vertex AI pipeline we need to create a cloud function with Pub/Sub Trigger. We start by creating a cloud function, we name our cloud function and set the Trigger type to 'Cloud Pub/Sub'.

After we save the settings in the code section we have configured the environment according to Python 3.7 and set the entry point as 'subscribe'.

```
import base64
import json
from google.cloud import aiplatform

PROJECT_ID = 'vlbaii-ss22-uc-3-development'
REGION = 'us-central1'
PIPELINE_ROOT = 'gs://vertexai_pipeline_vlba'

def subscribe(event, context):
  """Triggered from a message on a Cloud Pub/Sub topic.
  Args:
        event (dict): Event payload.
        context (google.cloud.functions.Context): Metadata for the event.
  """
  # decode the event payload string
  payload_message = base64.b64decode(event['data']).decode('utf-8')
  # parse payload string into JSON object
  payload_json = json.loads(payload_message)
  # trigger pipeline run with payload
  trigger_pipeline_run(payload_json)

def trigger_pipeline_run(payload_json):
  """Triggers a pipeline run
  Args:
        payload_json: expected in the following format:
          {
            "pipeline_spec_uri": "<path-to-your-compiled-pipeline>",
            "parameter_values": {
              "greet_name": "<any-greet-string>"
            }
          }
  """
  pipeline_spec_uri = payload_json['pipeline_spec_uri']
  parameter_values = payload_json['parameter_values']
```

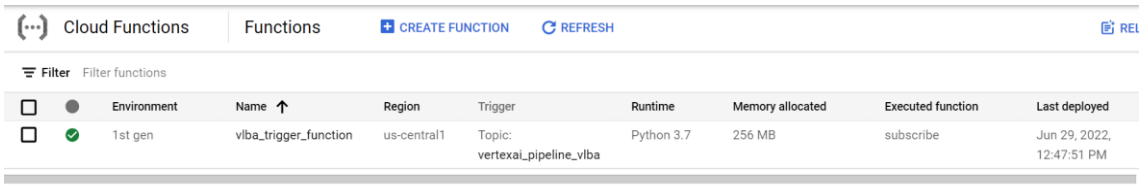
```
# Create a PipelineJob using the compiled pipeline from pipeline_spec_uri
aiplatform.init(
    project=PROJECT_ID,
    location=REGION,
)
job = aiplatform.PipelineJob(
    display_name='hello-world-pipeline-cloud-function-invocation',
    template_path=pipeline_spec_uri,
    pipeline_root=PIPELINE_ROOT,
    enable_caching=False,
    parameter_values=parameter_values
)

# Submit the PipelineJob
job.submit()
```

In the above-mentioned code we have set PROJECT_ID as ‘vlbaii-ss22-uc-3-development', REGION as 'us-central1' and PIPELINE-ROOT as 'gs://vertexai_pipeline_vlba'.

We need ‘google-api-python-client’ and ‘google-cloud-aiplatform’ for our code to work so we write following in ‘requirements.txt’ file.

```
google-api-python-client>=1.7.8,<2
google-cloud-aiplatform
```

Now we are all set to deploy. Once we deploy our function we will see the following instance in cloud functions.



4. Future Project Trends Forecast using Historical Data and Google Trends

This section outlines the details regarding the forecasting of future trends of projects in all the departments with the help of data obtained from SAP (stored in the BigQuery) along with Google Trends' historical data. Machine learning models were generated using VertexAI along with the Google Trends data stored in BigQuery. The ML models were created successfully and batch predictions for each model were made to generate forecast data for visualization and analysis. This section also includes the challenges faced during data preparation and model creation.

d. 4.1 Data Preparation (BigQuery and Google Trends)

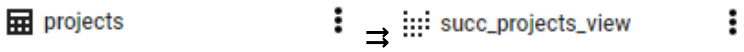
We required the use of views before we explored the provided dataset for visualization. We had to analyze every given schema to create meaningful connections in order to gain a general understanding of where the significant information should be retrieved from.

We created a few datasets in BigQuery in the us-central1 (Iowa) region (by default). They are as follows :

1. gtrends - To hold the historical data, views created for each department and data for forecast prediction (historical data with future timestamps).
2. gtrends_model_data - To hold the data exported from training all machine learning models.
3. gtrends_forecast_xxx - To hold the forecast data after batch prediction for the respective departments (xxx).

i. 4.1.1 View from Historical Data using BigQuery

To predict the future trends in the project, we considered only projects that were successfully completed. Following query was used to create the required view in BigQuery.




```
SELECT p.ID, p.description, p.RESNR as head_ID, p.RESNA as head_name, p.Business_Area
FROM `vlbaii-ss22-uc-3-development.VLBA_Group3.projects` p
where p.state='successful'
order by p.ID;
```

From the above view, we were able to query and get the details of successful projects for each department. The queries are as given below:

```
SELECT * FROM `VLBA_Group3.succ_projects_view` WHERE Business_Area = 'MPD'
SELECT * FROM `VLBA_Group3.succ_projects_view` WHERE Business_Area = 'CCR'
SELECT * FROM `VLBA_Group3.succ_projects_view` WHERE Business_Area = 'RSD'
SELECT * FROM `VLBA_Group3.succ_projects_view` WHERE Business_Area = 'MS'
```

We then created views for each department, from which we took distinct description values to form the data required to be given to the google trends search. The query for the distinct description is given below:

```
SELECT DISTINCT description FROM `gtrends.trends_mpd_historical`
SELECT DISTINCT description FROM `gtrends.trends_ccr_historical`
SELECT DISTINCT description FROM `gtrends.trends_rsd_historical`
SELECT DISTINCT description FROM `gtrends.trends_ms_historical`
```

ii.4.1.2 Google Trends Data

From the above views, we obtained necessary data to be used for the purpose of forecasting. We used the Google Trends web interface to obtain the data which depicts the interest over time of project trends worldwide for comparison (refer **Fig 15**).

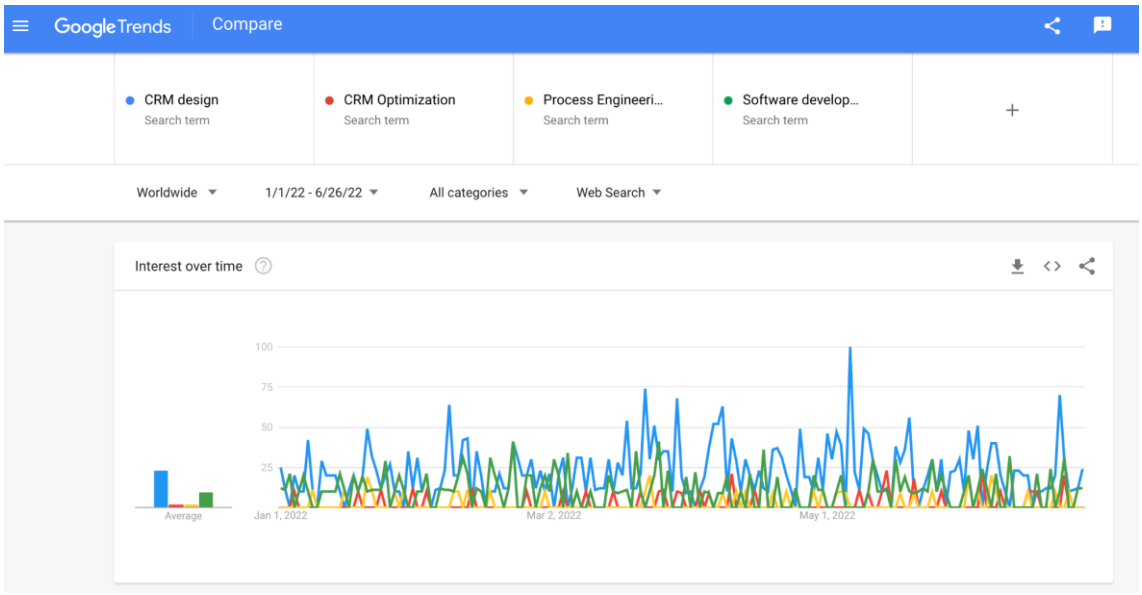


Fig 15: Google Trends Interest Over Time worldwide comparison data over a custom date range

For the MS department, we used the exact distinct description extracted from BigQuery view and for the other three departments, we used the pytrends library to extract related terms (rising or top) if the data for the exact description did not yield enough data for training the ML models. The pytrends code[4] was executed in Google Collab to extract related topics, it is as given below:

```
#install pytrends
!pip install pytrends

#import the libraries
import pandas as pd
from pytrends.request import TrendReq

#create model
pytrend = TrendReq()
#provide your search term
kw_list=['#values']
pytrend.build_payload(kw_list=kw_list)

# Related Topics, returns a dictionary of dataframes
related_topic = pytrend.related_topics()
related_topic.values()
```

Initially we obtained data on a monthly basis starting from 2004 till present to train the models. However, the data was not sufficient enough to train the models for forecasting. Hence we used the custom date range to obtain the data on a daily basis for a period of 5 years.

The ‘Timestamp’ and ‘Series Identifier’ columns are an essential component for model training in the Vertex AI AutoML tabular forecasting model. The date field we received from Google Trends was of the format YYYY-MM-DD, therefore we had to convert the date field into timestamp by altering the format to YYYY-MM-DD 00:00:00 UTC with help of excel. We also added the Series Identifier column into the data sets with values 1 to 4 for each department respectively. This data was stored into the already created BigQuery dataset (gtrends) as trends_xxx for each department(xxx). Future timestamps and Series Identifier were added to the historical data for the purpose of batch prediction and this new data was added to the gtrends dataset as xxx_forecast_data for each department(xxx). This data is then used for successful model creation and generation of forecast data.

e. 4.2 Machine Learning Models (VertexAI)

We enabled the VertexAI API to be used for machine learning model creation. It is a single environment to complete all of the ML work. We used AutoML mode provided by Vertex AI to create models, faster, with very few input. The steps carried out for model creation are described below.

i. 4.2.1 Datasets

Initially we need to fetch the data from BigQuery for creating the ML model using the AutoML training method. In the Vertex AI dashboard we navigated to ‘Datasets’ and chose the option to create a dataset (refer **Fig 16**).

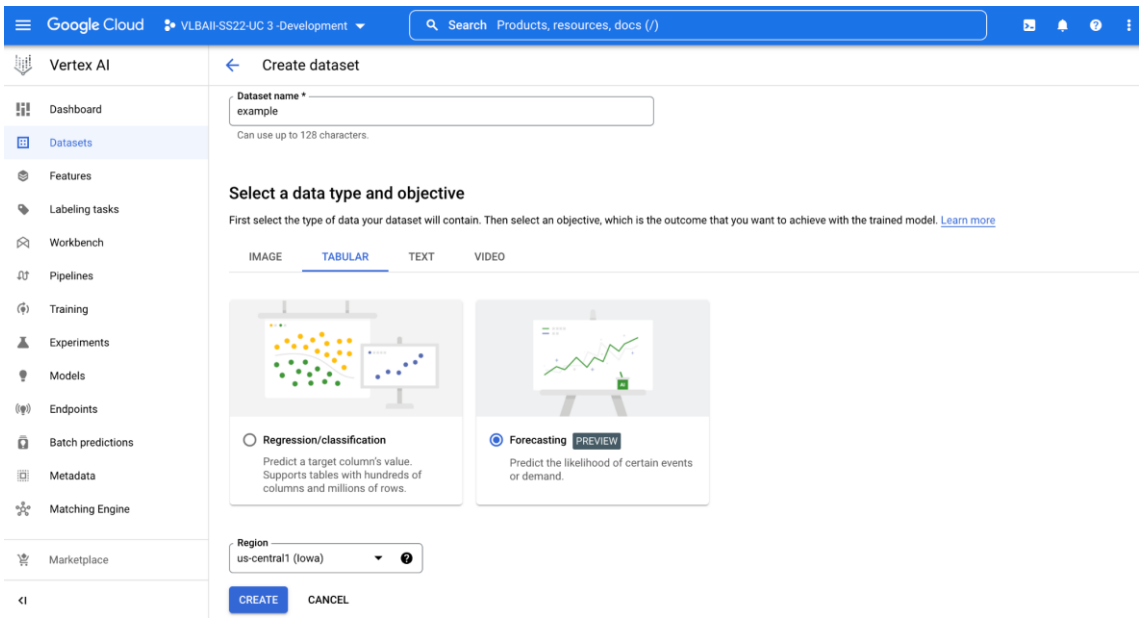


Fig 16: Creating Dataset in Vertex AI for AutoML Tabular Forecasting

In the ‘Create dataset’ page we added the unique dataset name and then we selected the ‘Forecasting’ in ‘Tabular’ as objective and data type respectively. We also have to set the same region as the one we chose for the bigquery during creation of the dataset (as default was set to us-central1 (Iowa)).

The data needs to be imported from gtrends dataset in BigQuery for each department. We have to select the data source as 'Select a table or view from BigQuery' and then browse and select the path and continue (refer **Fig 17**). The dataset will be added to the Vertex AI.

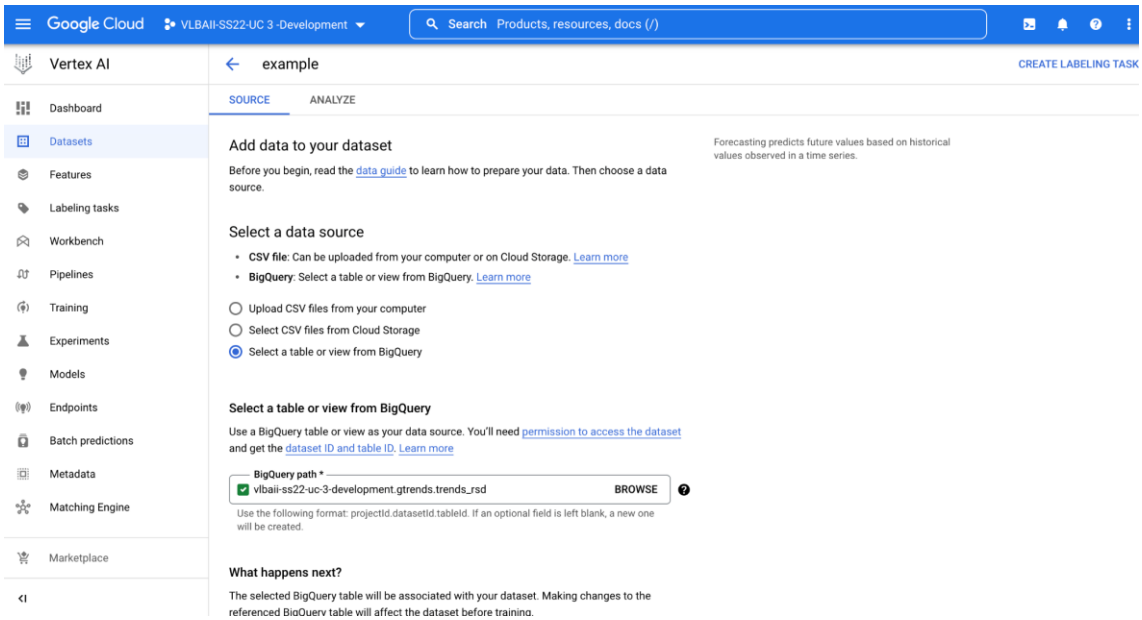


Fig 17: Importing RSD department dataset from BigQuery to Vertex AI

We created four datasets (trends_xxx) for each department (xxx) for the purpose of generating the respective ML models (refer **Fig 18**). Each dataset consists of a Timestamp column, a Series Identifier column and four other project descriptions or related topic columns.

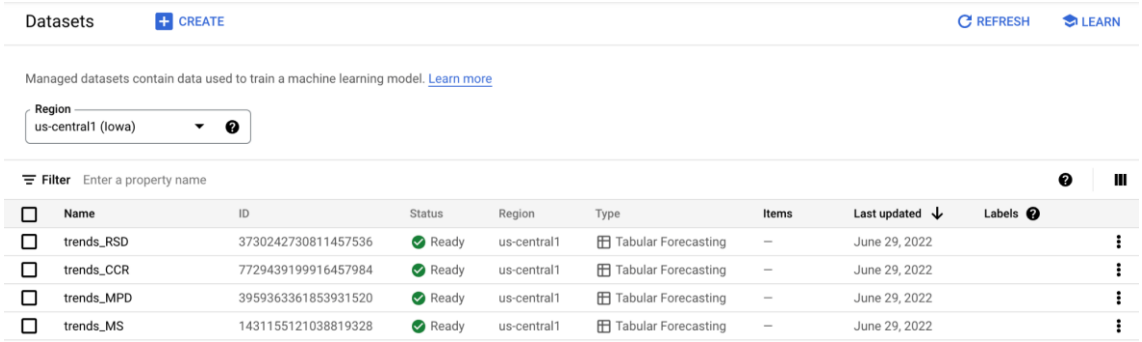


Fig 18: Imported Dataset of each department in Vertex AI

ii. 4.2.2 Training

In the Vertex AI dashboard, we navigated to the Training tab and clicked on the create button to start the training of the machine learning models. Here we created models for every project topic in respective departments i.e., we trained 4 models for each department, which sums up to 16 ML models in total which corresponds to the 16 project topics that are considered for forecasting the future trends.

The training consists of the following four steps:

1. **Training Method :** In this step we have to select the dataset upon which we need to train the ML models and for all of them we selected AutoML as the model training method. Then we continue to the next step.
2. **Model Details :** Here we gave a unique name to the model (ex: trends_RSD1) for better understanding and selected one of the target columns we need prediction for. We also set the 'Series Identifier' and 'Timestamp' Column from the dataset. In Forecasting configuration, we set the 'Data granularity' to 'Daily' which is the granularity level of the timestamp column. We also set the 'Forecast horizon' and 'Context Window' to the size of 180 days to receive predictions for the future 180 days. Additionally we exported the test dataset to BigQuery to the gtrends_model_data dataset for each model (ex: export_rsd1).
3. **Training Options:** In this section, for the project topics other than the one used for prediction, we changed the option to 'Not Available' for forecasting and under the 'Advanced options' we selected 'MAE' as 'Optimization objective' to view extreme values as outliers with less impact on the model.
4. **Compute and Pricing :** In this step, we set the Budget to be around one node hour. This is the minimum amount of time for which one can train the model.

An email was sent to notify the completion of training of each model. It is necessary to have a minimum of 1000 rows of training examples in the dataset in order to train a model. The training completion is marked with 'Finished' status (refer **Fig 19**).

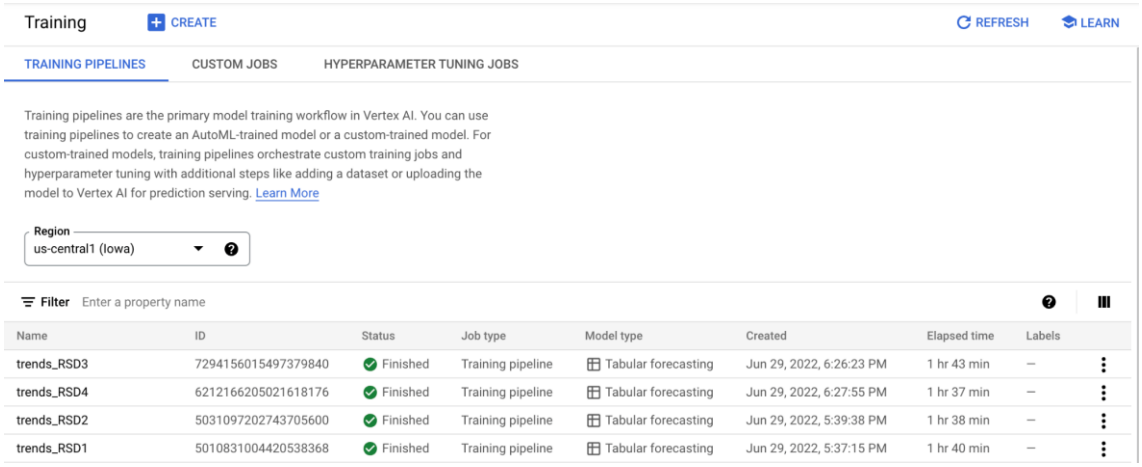


Fig 19: Example of RSD Department Models with training Finished status

iii. 4.2.3 Models

Now we can navigate to the 'Models' section in VertexAI to see the list of completed models.

f. 4.3 Batch Predictions (VertexAI)

After successful model training completion, we can now start with the 'Batch predictions' in Vertex AI to generate the predicted value and store it in BigQuery for the visualization of the forecast along with historical data from Google Trends.

We then navigated to the 'Batch predictions' in Vertex AI and clicked on 'Create' to start with batch predictions. In the 'New batch prediction' window, we gave a unique name to each batch prediction (ex: trend_RSD1_forecast) and then chose the model (ex: trends_RSD1) which is one of the trained ML models. We set the version of the ML model to be the default version (version 1).

In the next step, we needed to provide the source (BigQuery) which is the data that consists of historical data with future timestamps along with series identifier values and feature values as null so the prediction could be performed for those timestamps. In our case, it was stored in gtrends dataset in BigQuery as xxx_forecast_data (ex: rsd_forecast_data) for each department (ex: RSD). The batch prediction output was saved into gtrends_forecast_xxx (ex: gtrends_forecast_rsd) dataset in BigQuery for the visualization part.

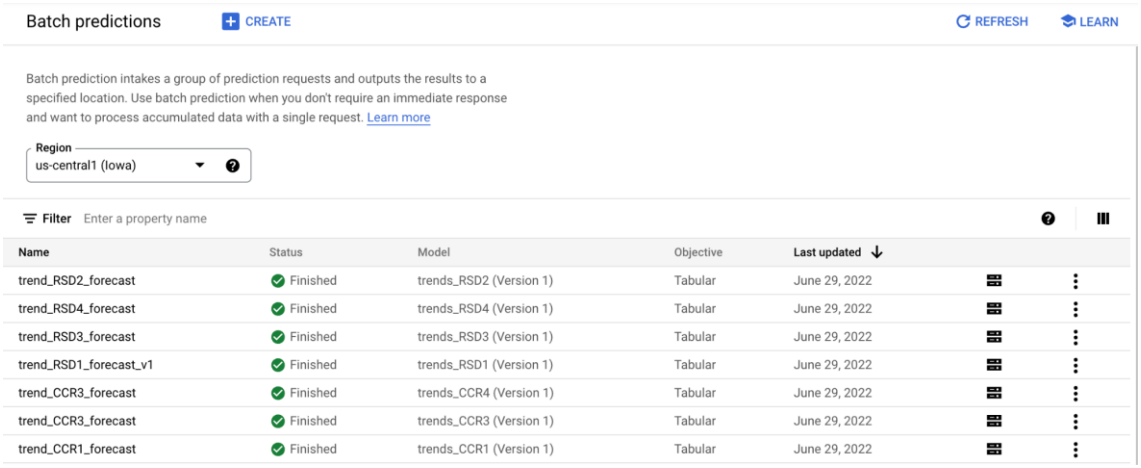


Fig 20: Example of RSD Department Batch predictions with Finished status

After batch predictions, we can see that the data is stored under the gtrends_forecast_xxx for the respective ML models of each department (xxx). The completed training pipelines can be viewed under the 'Batch predictions' tab in Vertex AI dashboard (refer **Fig 20**) .

g. 4.4 Visualization and Data Analysis (Google Data Studio)

The forecast data was successfully saved into the BigQuery after batch predictions using Vertex AI. This data along with the historical, old and future timestamp data were utilized to visualize the forecast data in the Google Data Studio .

From the BigQuery gtrends_forecast_xxx (ex: gtrends_forecast_rsd) dataset, we exported the prediction data (ex: trends_RSD1_forecast) to Google Data Studio. For this, we selected one of the prediction data and on doing so, the schema details of the forecast data appear towards the right-hand side. Under the 'Export' option, we selected the option 'Explore with Data Studio' which will take us into the Google Data Studio web interface with the forecast data being already imported.

We imported the rest of the data required (mentioned above) through the 'Add data' button in Google Data Studio from BigQuery. The imported files are:

1. The prediction data of all features (ex: trends_RSD1_forecast)
2. The historical data - trends (ex: trends_RSD)
3. The historical data along with future timestamps with null feature values. (ex: rsd_forecast_data)

We created four data visualization reports (for each department) with three pages which are explained in the following sections. Time series charts were used for data visualization. The pages in each report are named as follows:

1. **Forecast** : Prediction data visualization
2. **Historical** : Historical data visualization

3. **Forecast + Historical** : Combination of both for a better understanding of the predicted and historical data.

i. 4.4.1 Forecast Data

For the RSD department, we used the predicted data (trends_RSD1_forecast, trends_RSD2_forecast, trends_RSD3_forecast and trends_RSD4_forecast) to create the forecast in the Time series chart. We connected these four data using the 'blend Data' option in Data Studio.

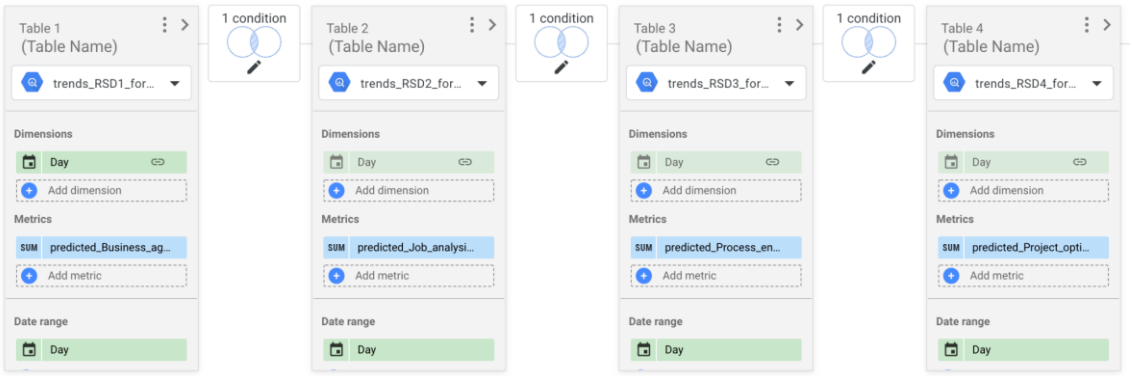


Fig 21: Blend Data configuration for Forecast Data (RSD)

In the blend data window, we joined the four data using the inner join condition with 'Dimensions' as 'Day' column (timestamp) and the 'Metrics' were the predicted values in feature columns (refer **Fig 21)**. In each data or table we had around 180 record count as we set the context window length to be 180 days, the predictions were made for the future 180 days. The 'Date Range' was again set as the 'Day' column for the graph to form.

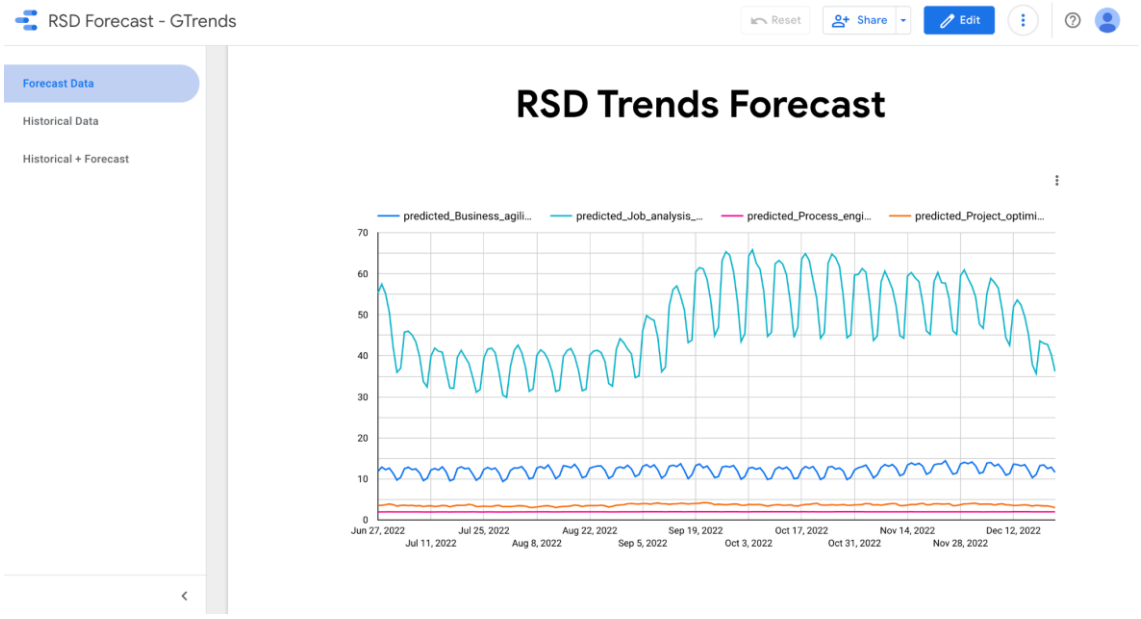


Fig 22: Forecast Data of Trends (RSD)

After saving the blend data configuration, we got the above (refer **Fig 22**) Time series chart depicting the forecast for July 2022 to December 2022. Since we used MAE during the model training in Vertex AI, the prediction became less impacted by the extreme outliers which resulted in simpler prediction charts.

ii. 4.4.2 Historical Data

For the Time series chart of Historical Data of the RSD department, we set the 'Data Source' as the trends_rsd data, 'Dimension' was set to the Day column and all the four features were added as the 'Metric' which resulted in the following chart (refer **Fig 23**).

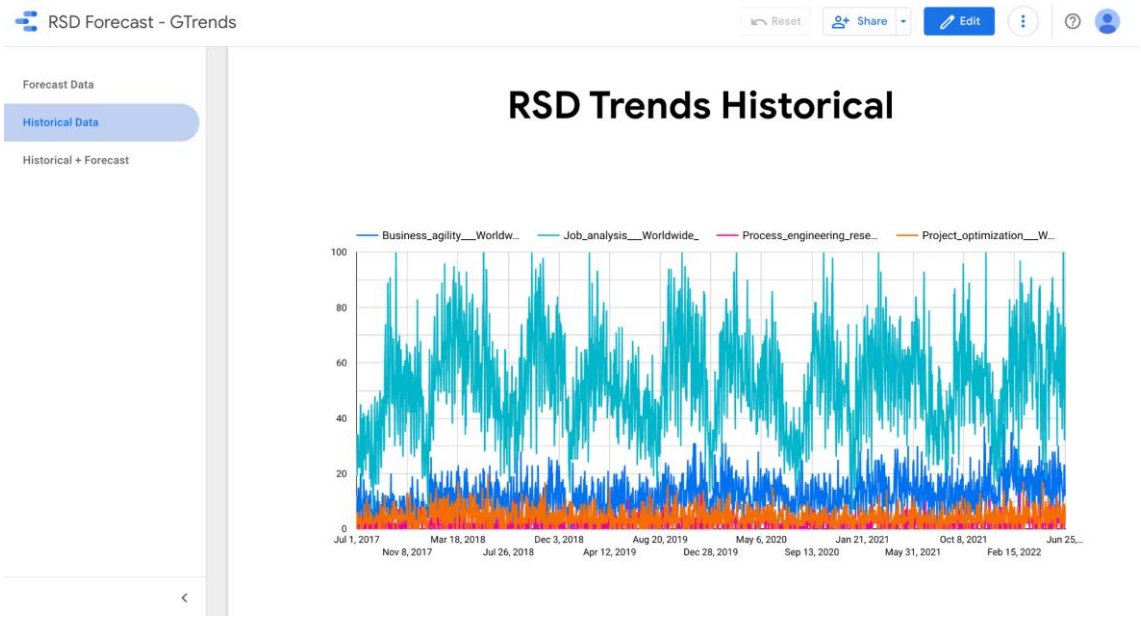


Fig 23: Historical Data of Trends (RSD)

The above chart represents the historical data (with timestamp and series identifier) from Google Trends which was stored into BigQuery. This is the data which was used for training the ML models.

iii. 4.4.3 Forecast with the Historical Data

This page consists of the combination of both the data used for forecasting (ex: rsd_forecast_data - consists of future timestamp) and the predicted data for each feature in RSD department (ex: trends_RSD1_forecast, trends_RSD2_forecast, trends_RSD3_forecast and trends_RSD4_forecast) through the outer join connection (refer **Fig 24**).

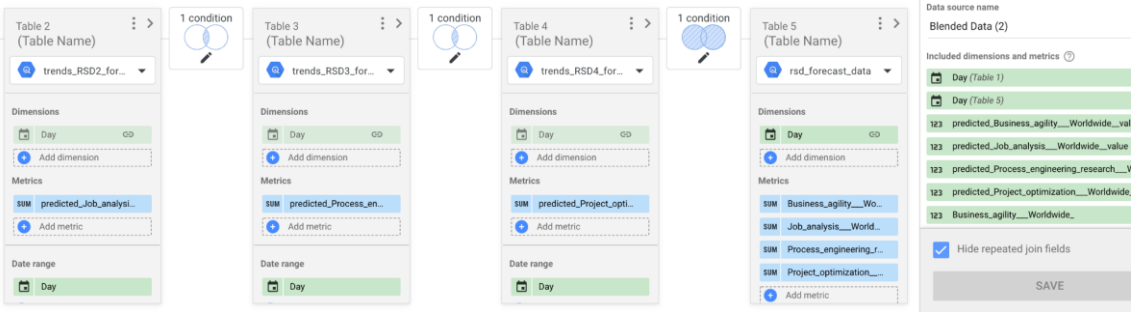


Fig 24: Blend Data configuration for Forecast and Historical Data (RSD)

We connected these data using the 'blend Data' option in Data Studio. Through the above configuration we plotted the predicted values of each feature and the historical data of each feature in the Time series chart. We also set a custom date range from January 2022 to December 2022 to show the historical and predicted values in 6 months each (refer **Fig 25**).

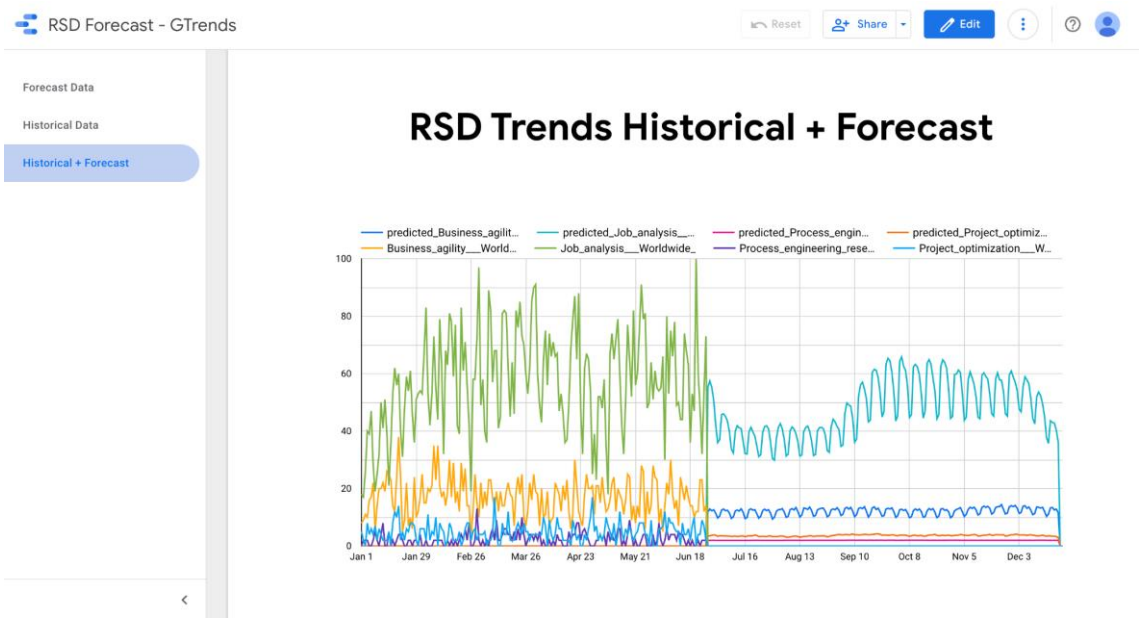


Fig 25: Forecast Data with the Historical Data of Trends (RSD)

h. 4.5 Workflow Diagrams

This section showcases the various workflow diagrams for Future Project Trends Forecast using Historical Data and Google Trends.

i. 4.5.1 Data Preparation

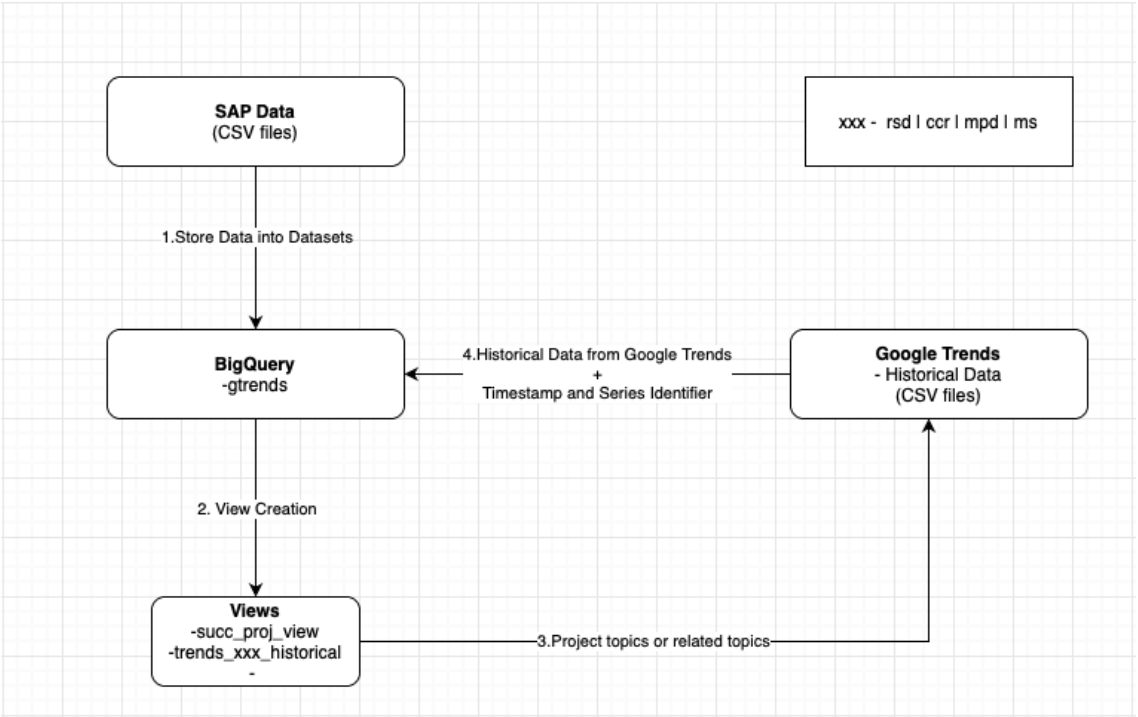


Fig 26: Data Preparation Workflow

ii. 4.5.2 Machine Learning Models and Batch Predictions

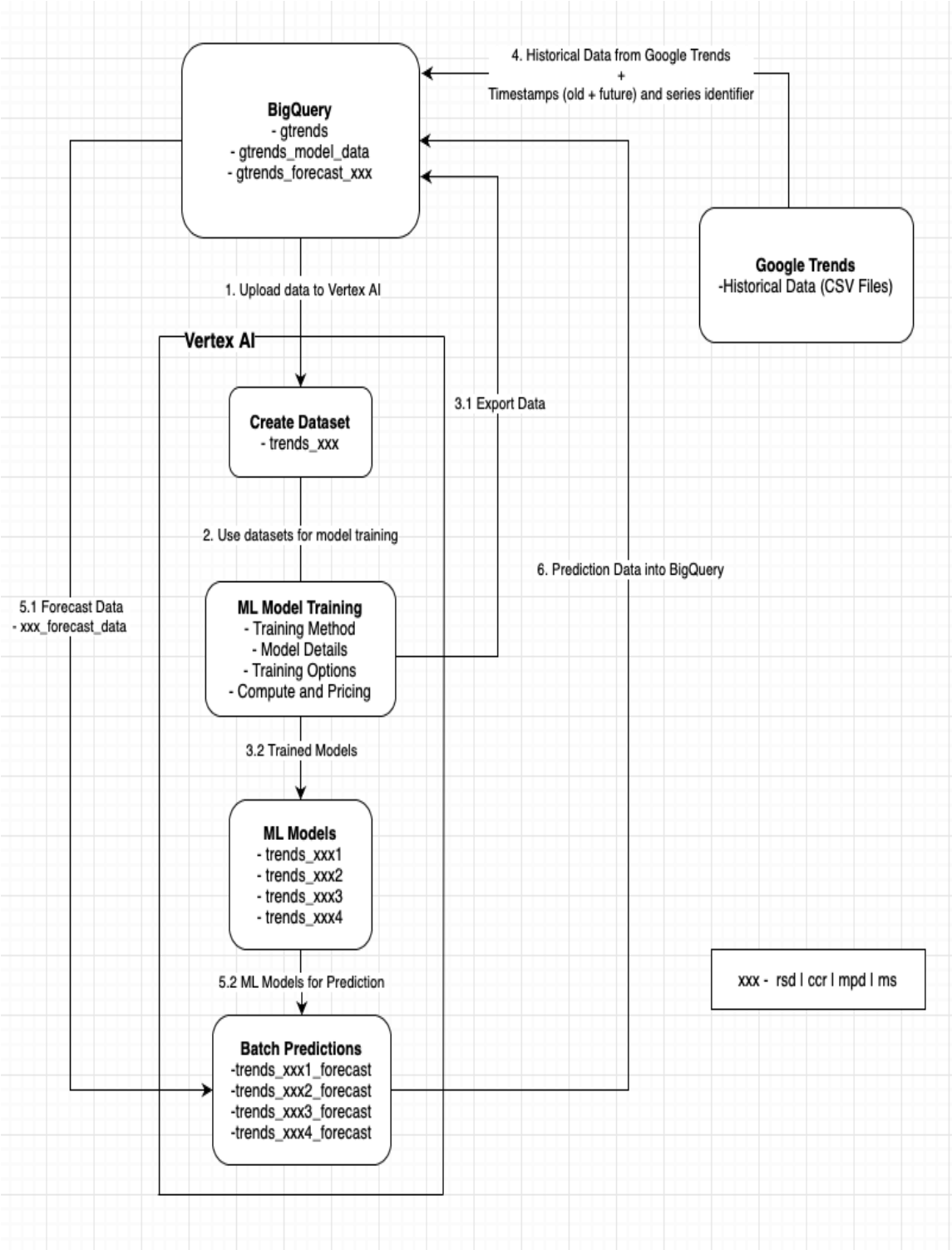


Fig 27: Vertex AI Workflow

iii.4.5.3 Visualization

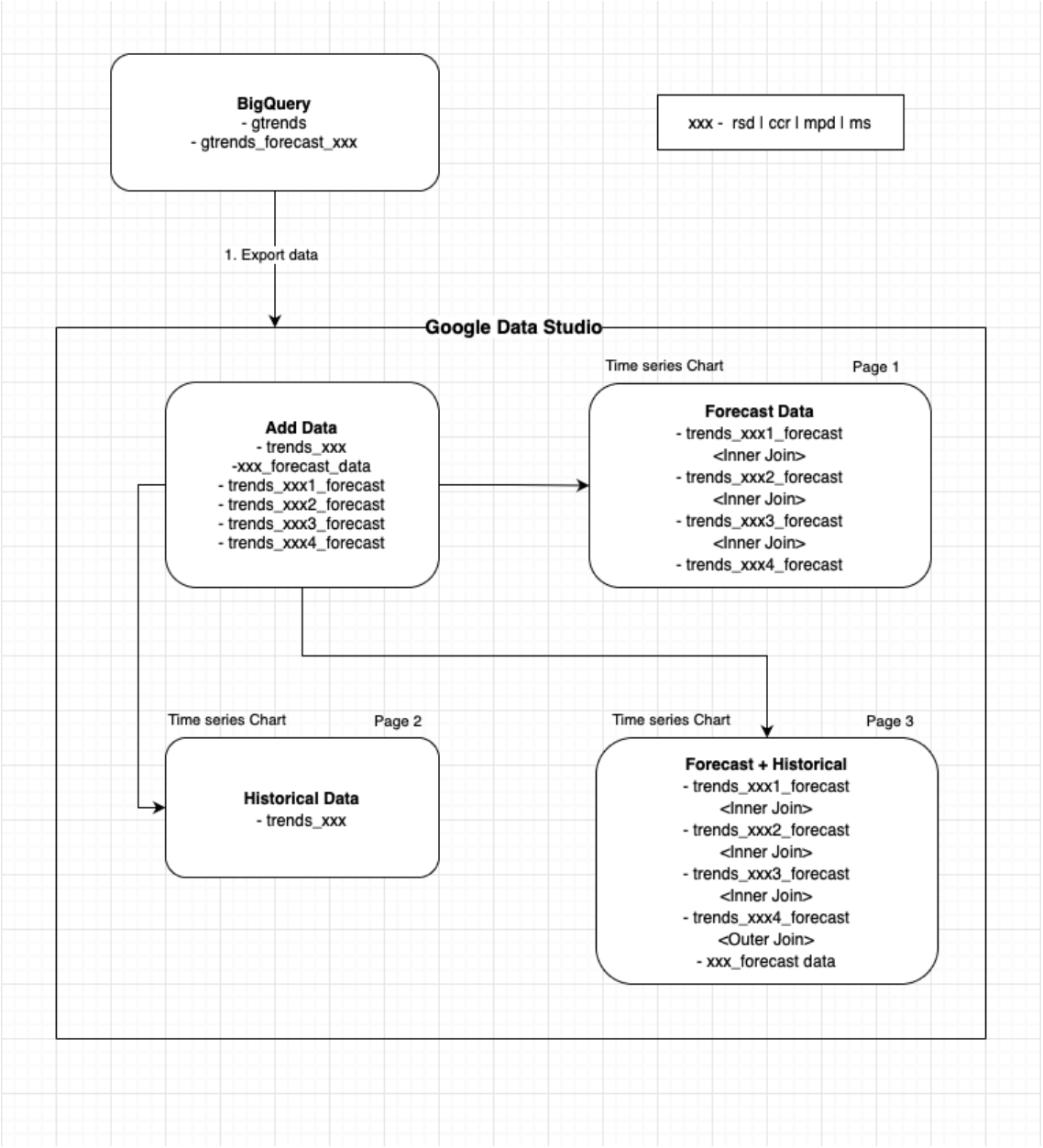


Fig 28: Google Data Studio Visualization Workflow

i. 4.6 Challenges and Limitations

This section includes all the challenges faced during the implementation of this task using various platforms such as Google Trends, Vertex AI and Google Data Studio.

i. 4.6.1 Google Trends

1. We can only compare up to 5 topics in Google Trends due to which we were not able to add more related topics.
2. We could only obtain limited data (222 rows) on a monthly basis from 2004 till date which was not sufficient to train ML models in VertexAI (which requires a minimum of 1000 rows). Hence, we had to obtain the data on a daily basis to train the model by using a custom date range over 6 months.

3. Monthly data is relevant for long term prediction, but due to less data, we had to utilize daily data which is only sufficient for short term behavior analysis. So the forecast data fails to capture long-run trends.
4. We were not able to get sufficient data for the given project descriptions. Even though we tried to find related project topics, most of them were not relevant to our objective.
5. While trying to find related topics similar to the given project topics, we were unable to utilize the Python pytrends library due to the limitation of input search keywords being at most one.
6. Google Trends displays the relative score of data and it is not possible to get the absolute value.

ii. 4.6.2 Vertex AI

1. We need a minimum of 1000 rows in the dataset to train a model.
2. Since we had around 16 models to train, the quota exhaustion of concurrent jobs in the Vertex AI model training pipeline led to more training hours and also failure of some models which had to be trained again.
3. After the model training, the data types of all the features including timestamp and series identifier were converted to string values and only the predicted value had a data type of float. This caused an issue during data visualization in Google data studio as the 'Day' column was read as a String, and not a timestamp value.

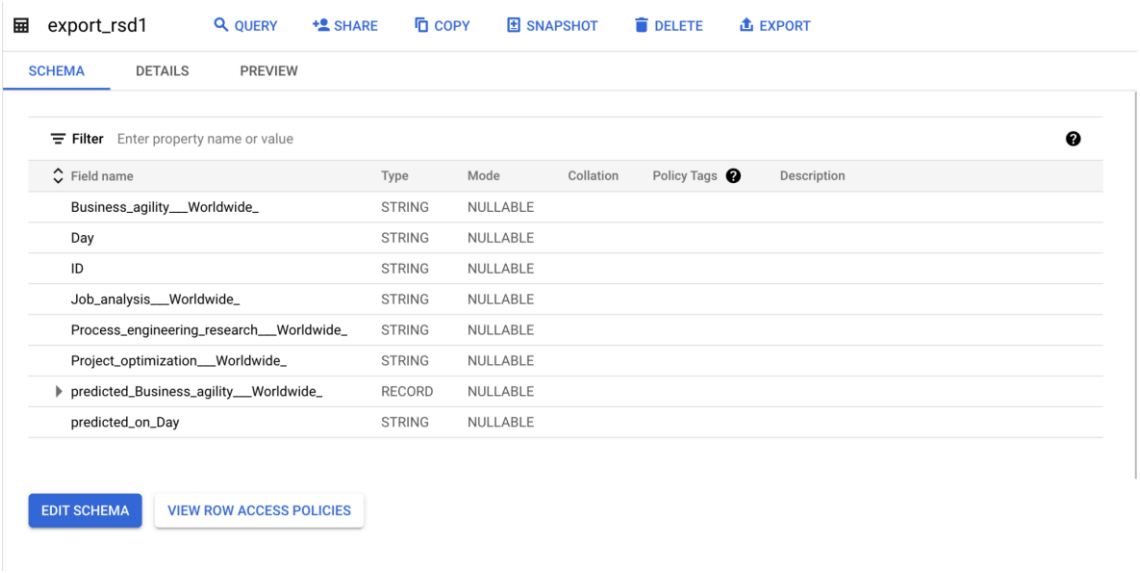


Fig 29: Data inconsistency in ML models after training

To overcome this challenge, we had to download the batch predicted data as CSV files using the BigQuery 'Export to sheets' option. We then uploaded these CSV files into the gtrends_forecast_xxx containing the inconsistent predicted data(in terms of data type) as trends_xxx_forecast for each feature prediction.

4. Similar to the ML model pipeline, the Batch predictions pipeline posed trouble due to concurrent job quota exhaustion which led to failure of several batch predictions which then had to be created again.

iii.4.6.3 Google Data Studio

1. We were unable to use Time series for one of the visualizations with the error being 'Too many rows', but it worked for other models without any error.
2. After exporting data from BigQuery into Google Data Studio, we had to save and close the report and then reopen again to add other relevant data since we were unable to add any other data before saving it.

j. 4.7 Google Data Studio Report

This section consists of the Google Data Studio Report of all the forecasts for each individual department.

i. 4.7.1 MPD Department

1. Forecast Data

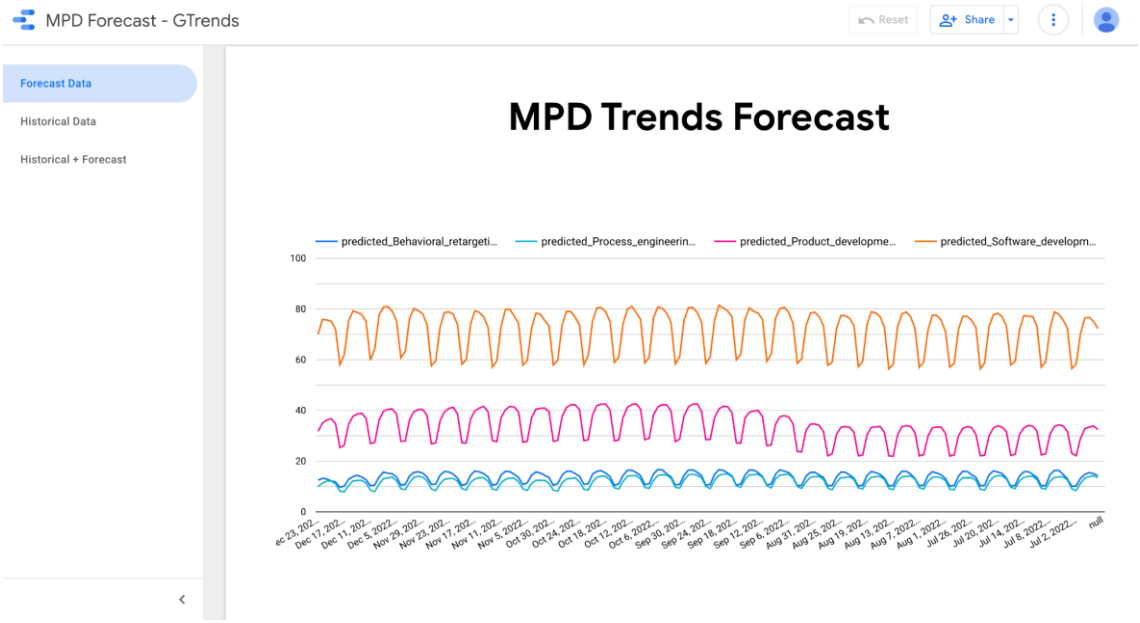


Fig 30: MPD Trends Forecast

2. Historical Data

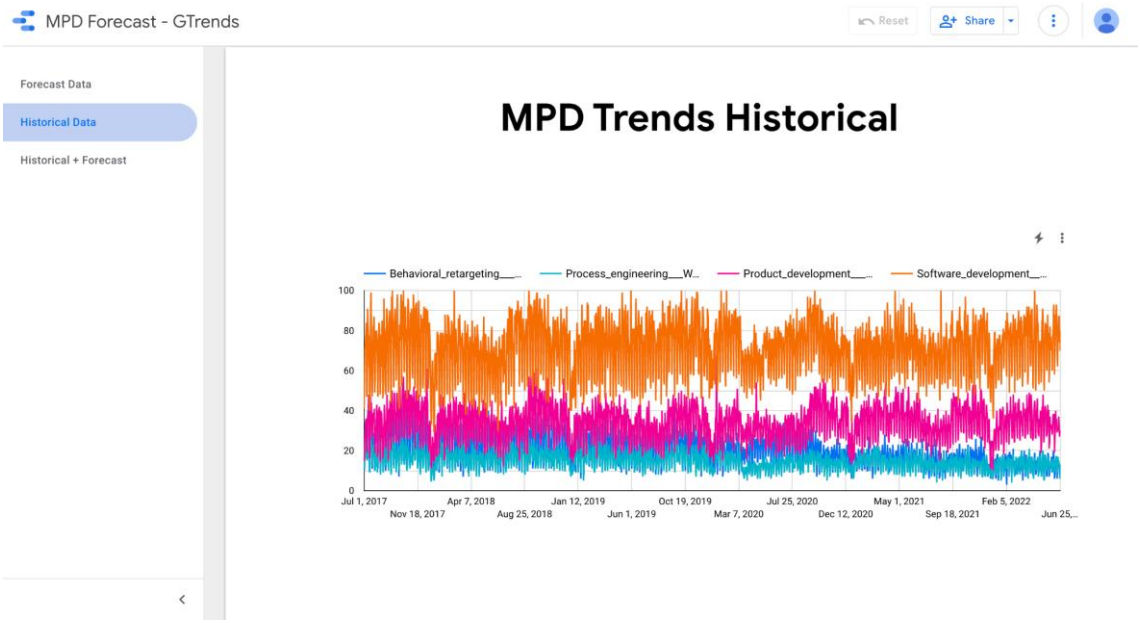


Fig 31: MPD Trends Historical

3. Forecast with Historical Data

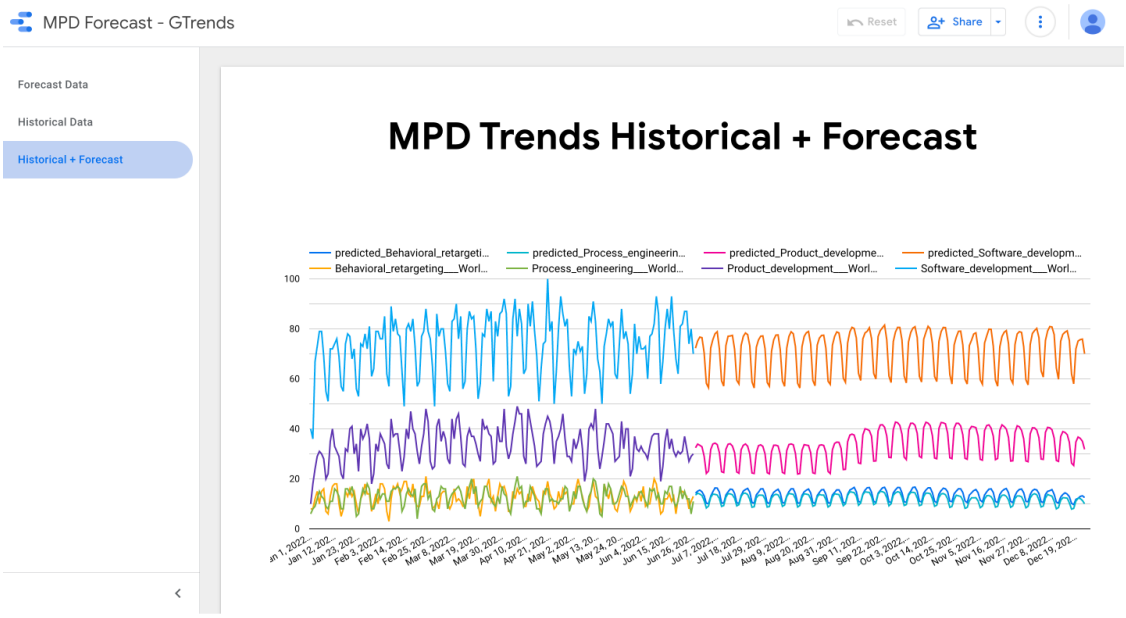


Fig 32: MPD Trends Historical + Forecast

ii. 4.7.2 CCR Department

1. Forecast Data

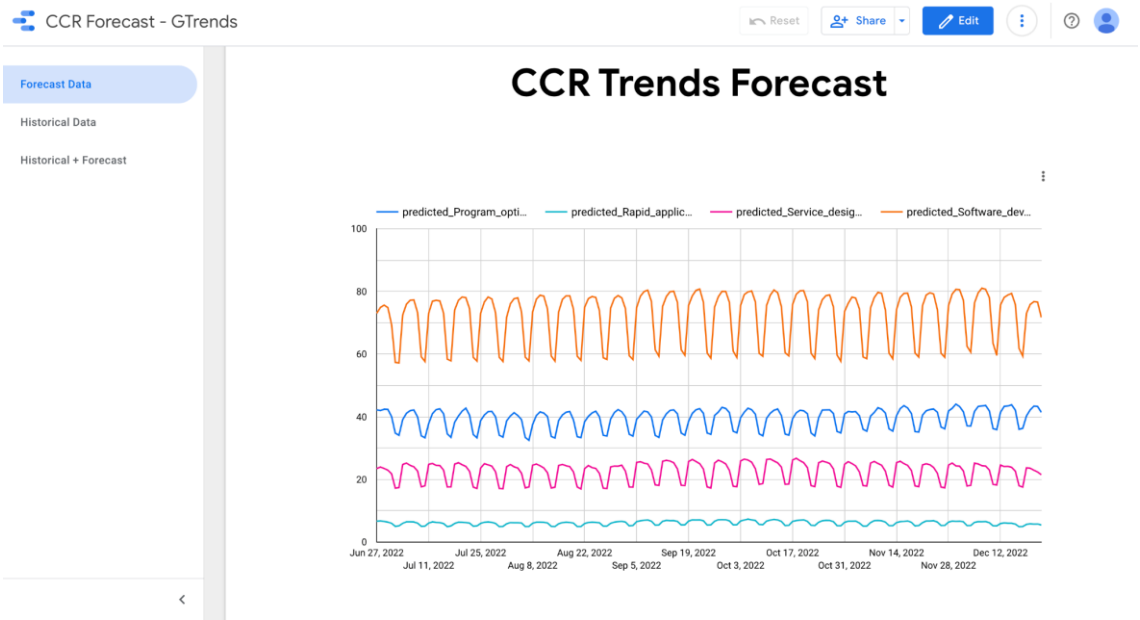


Fig 33: CCR Trends Forecast

2. Historical Data

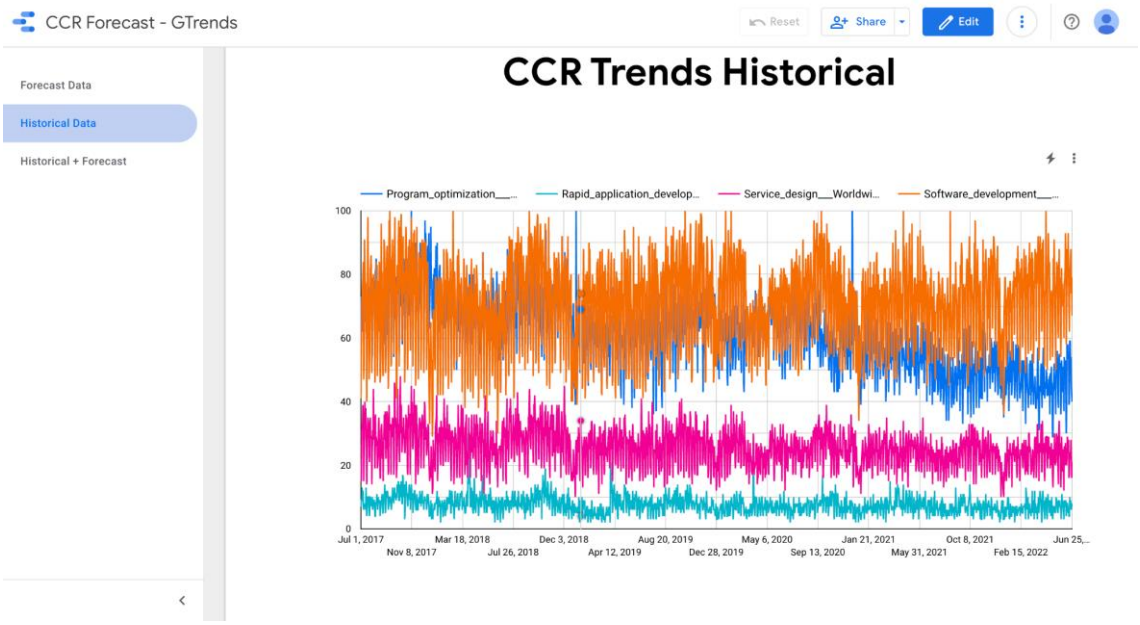


Fig 34: CCR Trends Historical

3. Forecast with Historical Data

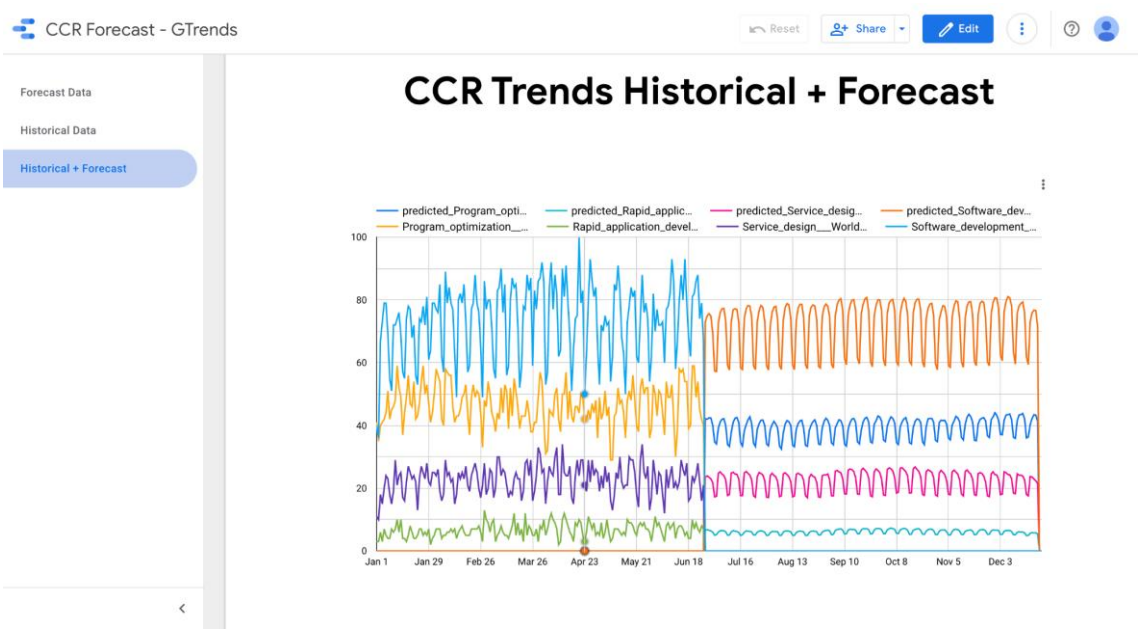


Fig 35: CCR Trends Historical + Forecast

iii. 4.7.3 RSD Department

1. Forecast Data

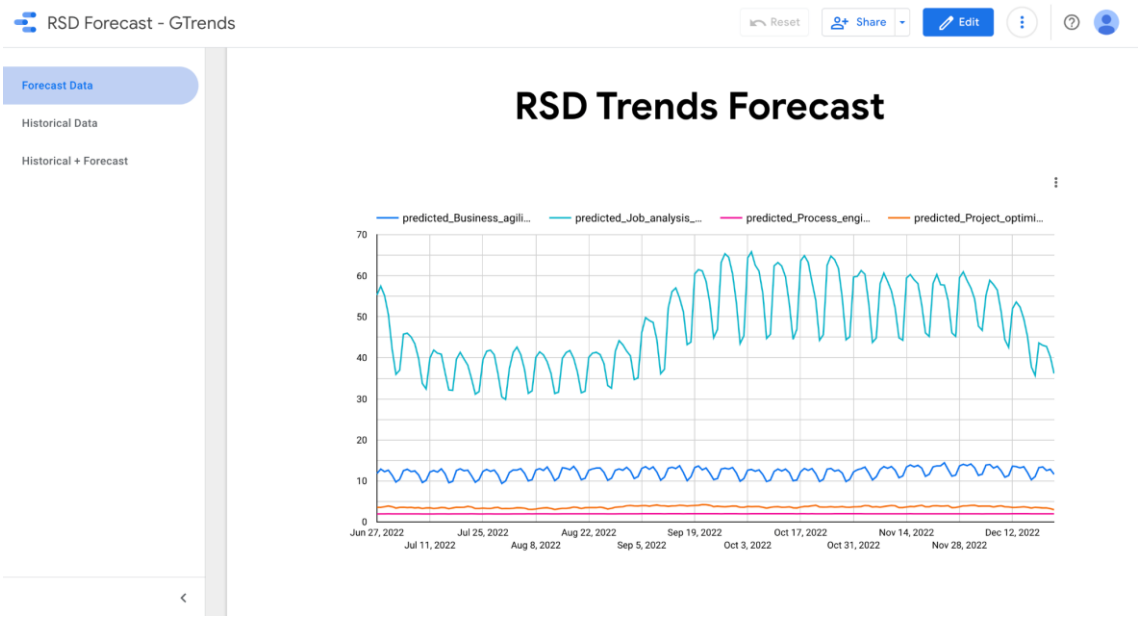


Fig 36: RSD Trends Forecast

2. Historical Data

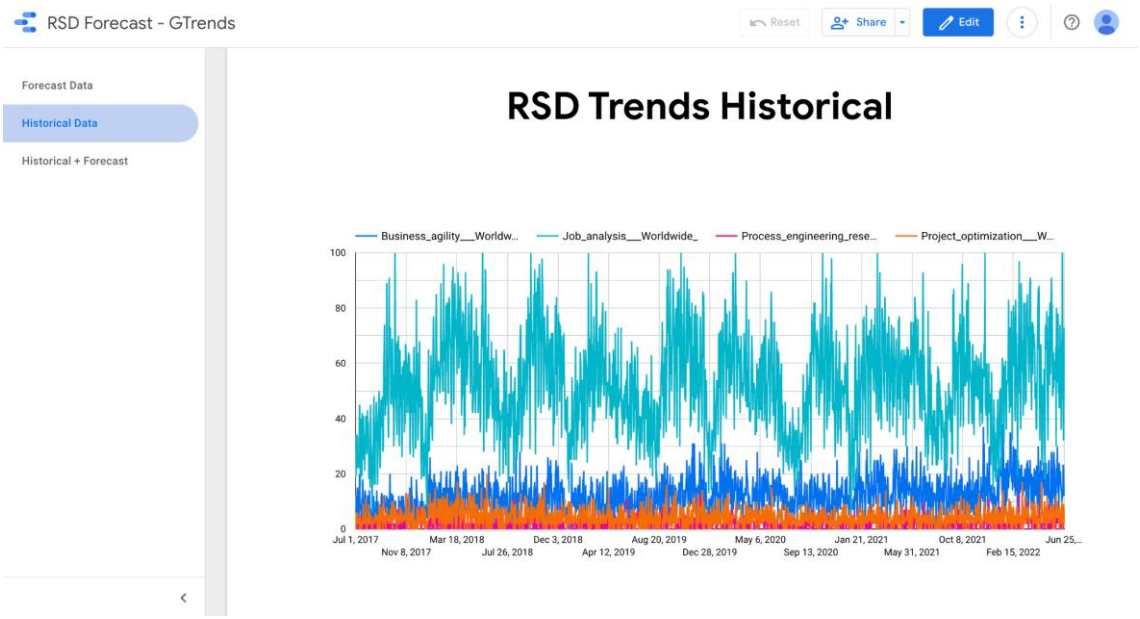


Fig 37: RSD Trends Historical

3. Forecast with Historical Data

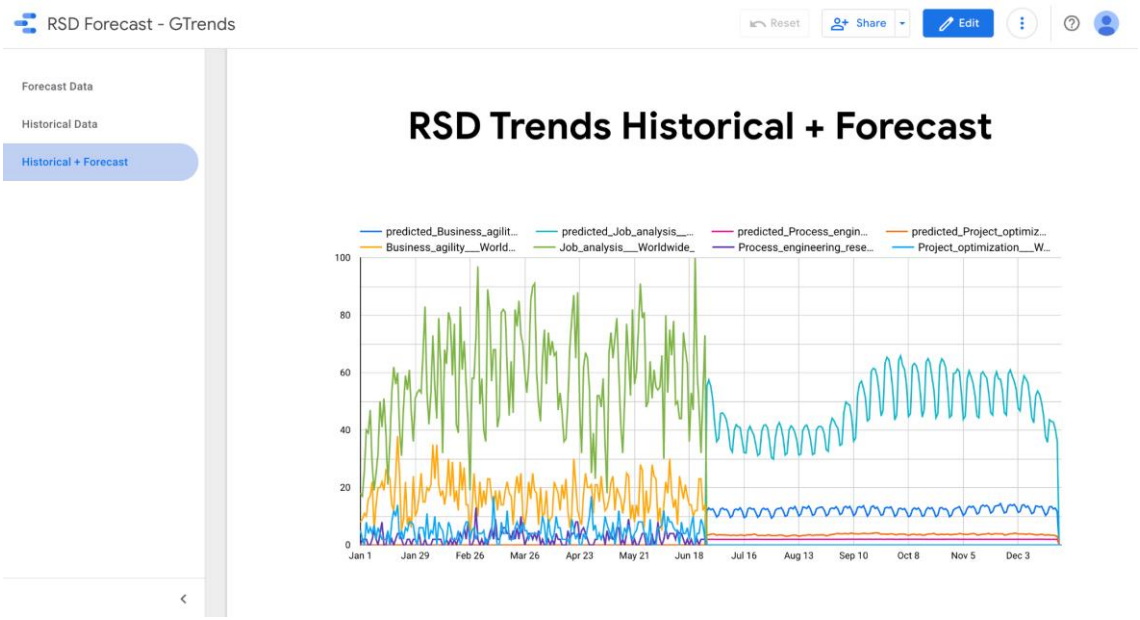


Fig 38: RSD Trends Historical + Forecast

iv. 4.7.4 MS Department

1. Forecast Data

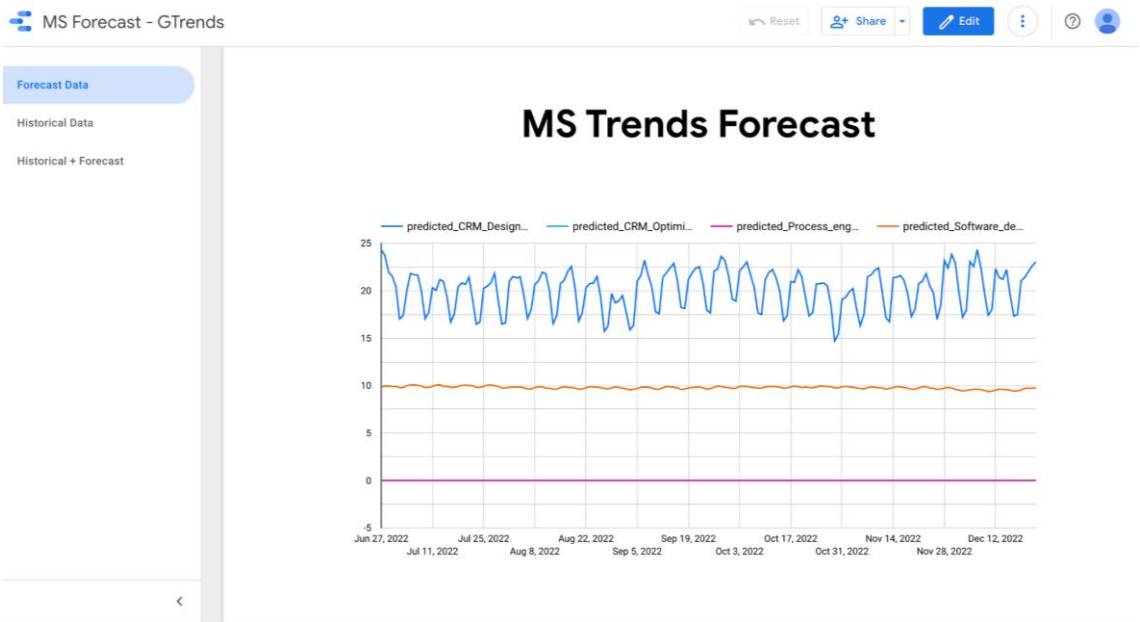


Fig 39: MS Trends Forecast

2. Historical Data

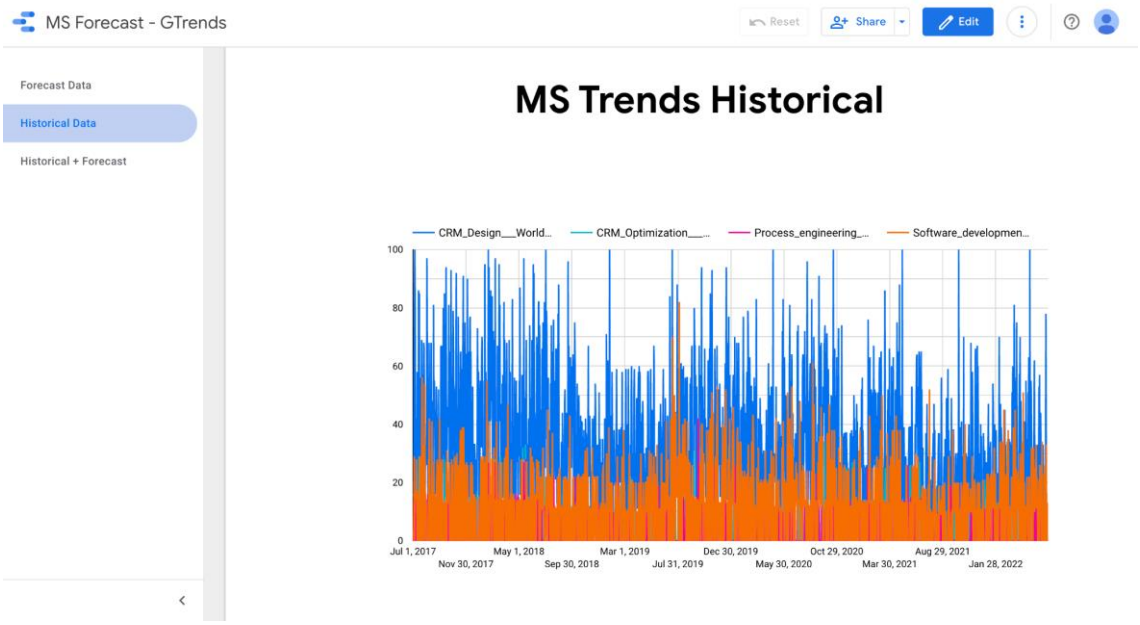


Fig 40: MS Trends Historical

3. Forecast with Historical Data

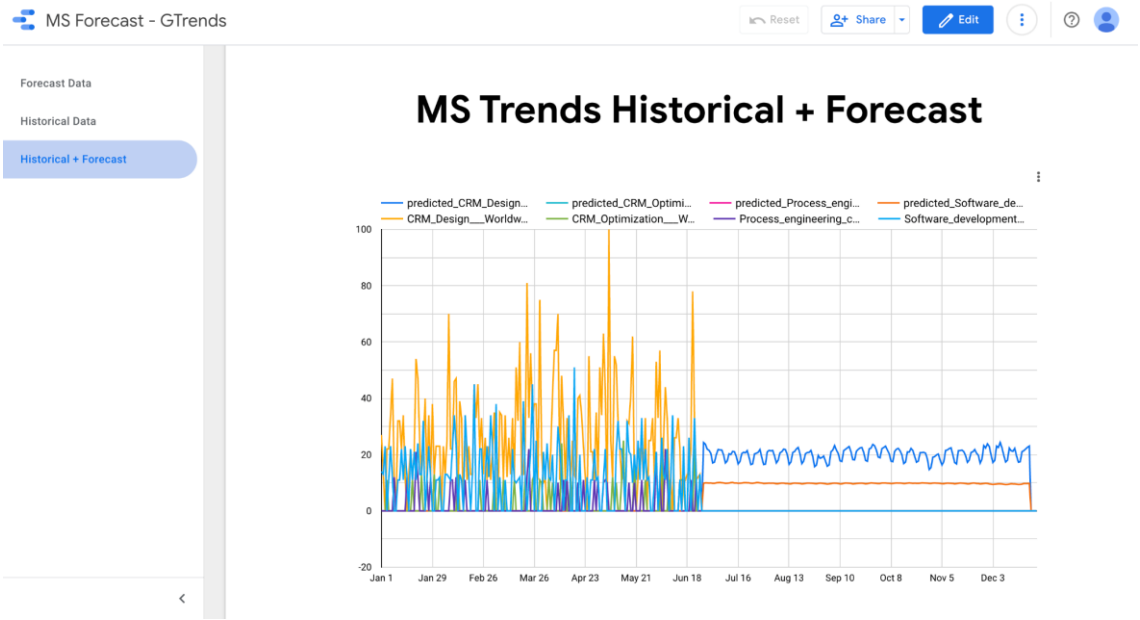


Fig 41: MS Trends Historical + Forecast

In conclusion, we successfully generated the reports for each department and data analysis was made possible through the help of Time series charts of historical, forecast prediction data and the combination of both.

7. Results

For the first task, we were able to utilize different aspects of Google Trends such as Historical data (Worldwide), related topics (rising and top) and Python pytrends[4] library within the limitations to extract valuable data and store it into BigQuery after necessary data alterations (timestamps and series identifier). We then successfully made ML models and Batch predictions using the Vertex AI AutoML Tabular Forecasting Model training. The predicted data was stored into BigQuery which was used later on to create visualizations with the help of Google Data Studio. The reports created were helpful in identifying future project topic trends for each department.

For the second task, in order to provide visualizations for recruiters, we initially had to derive meaningful views. This was done by using the in-built analytical features of BigQuery, wherein views were generated from the schema provided to us (via SQL commands). Resulting views were categorized by business area, employee characteristics, and project success score. These views were then used to analyze the current employee pool on the basis of academic titles and educational backgrounds, depending on the success rate of projects within different business areas. Furthermore, these success scores were used to train a machine learning model that predicted new candidates' success scores based on information like future position, education field and institution. Then a VertexAI pipeline was made for this model, which can be triggered using a Cloud Function service that subscribes to the Cloud Pub/Sub service.

For the third task, with the goal of gathering insights from university data for recruiting best talents, we created views ranking universities by their student’s success rate in projects for various business areas. We plotted these views listing the universities by success rate along with best faculties. The results showed that for each business unit, the top universities are varying. Recruiters can look at the list and decide which universities to collaborate with.

8. References

[1] How to build forecasting models with Vertex AI

https://www.youtube.com/watch?v=5-qjRpjdE5s

[2] What is BigQuery ML? (n.d.). Google Cloud.

https://cloud.google.com/bigquery-ml/docs/introduction

[3] Google Cloud Vertex AI accelerates machine learning. (n.d.). Google Cloud Blog.Retrieved June 24, 2022, from

https://cloud.google.com/blog/products/ai-machine-learning/google-cloud-vertex-ai-accelerates-machine-learning

[4] The Ultimate Guide to PyTrends: the Google Trends API (with Python code examples)

https://lazarinastoy.com/the-ultimate-guide-to-pytrends-google-trends-api-with-python/