Cloud Application Development Assignment

Serik Dinmukhamed 21B030922
03.11.2024

Table of Contents

Introduction

This report covers the tasks completed for Assignment 3, focusing on developing, deploying, and managing cloud-based applications using Google Cloud. The assignment was divided into three exercises, each emphasizing different areas of cloud application development: managing APIs with Google Cloud Endpoints, setting up a Cloud SQL database, and deploying machine learning models with Google Cloud AI Platform. Each task required the use of Google Cloud services, along with Python for development.

2. Exercise 1: Managing APIs with Google Cloud Endpoints

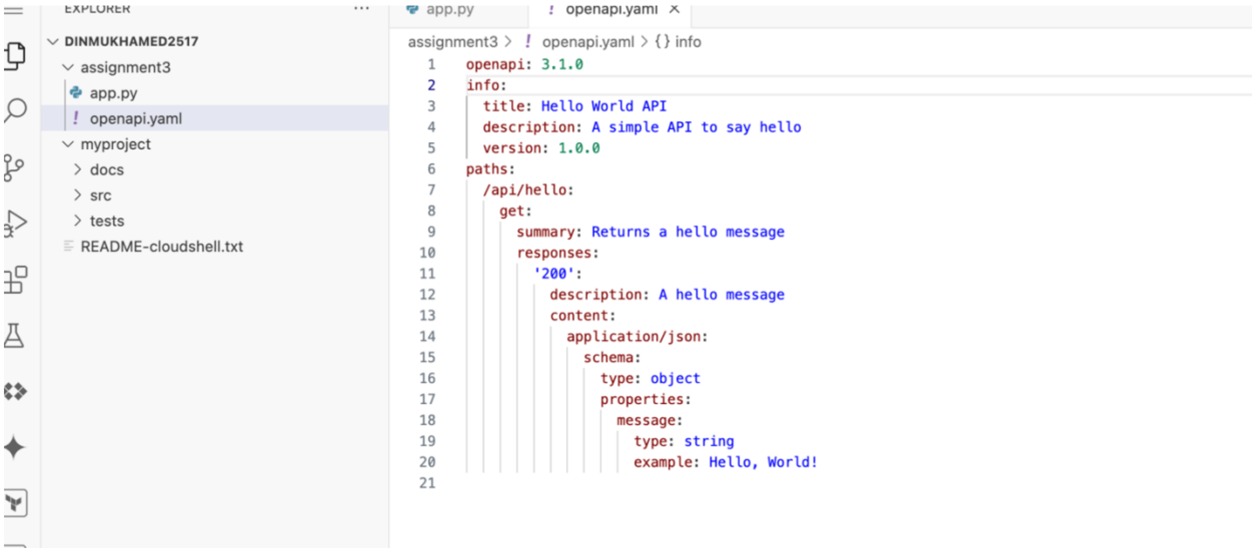
Objective

To create, deploy, and manage a REST API using Google Cloud Endpoints.

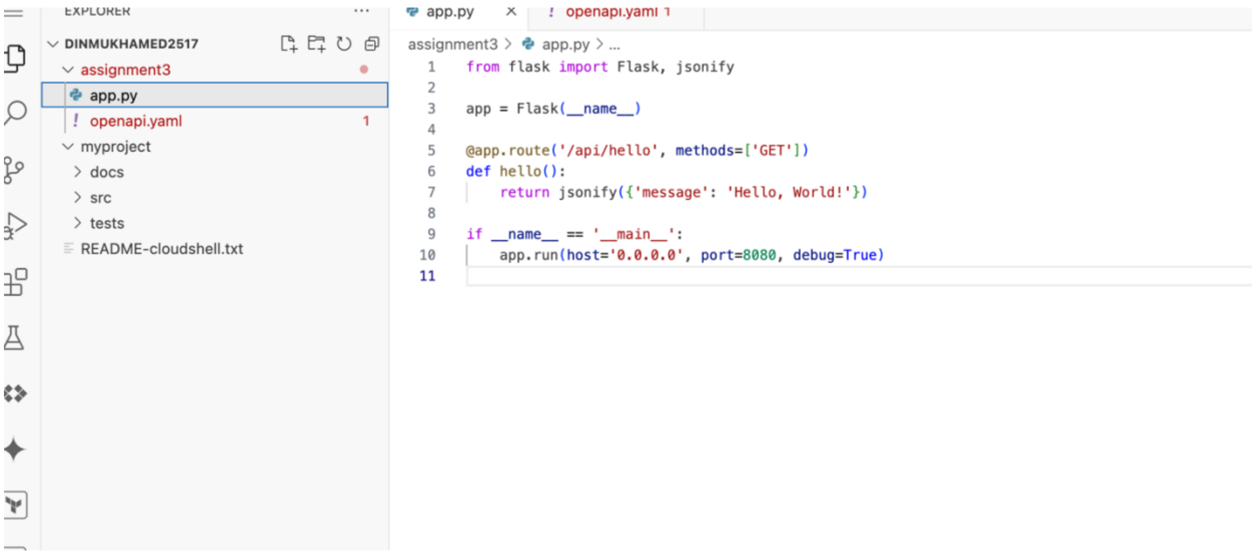
Steps and Implementation

API Development:

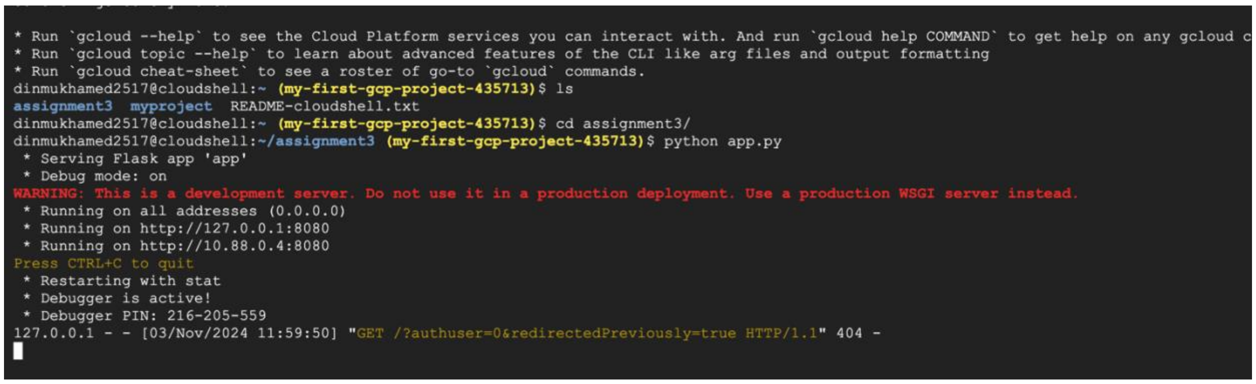
Created a simple REST API using Python and Flask. The API contains an endpoint (/api/hello) that returns a JSON response with a greeting message.



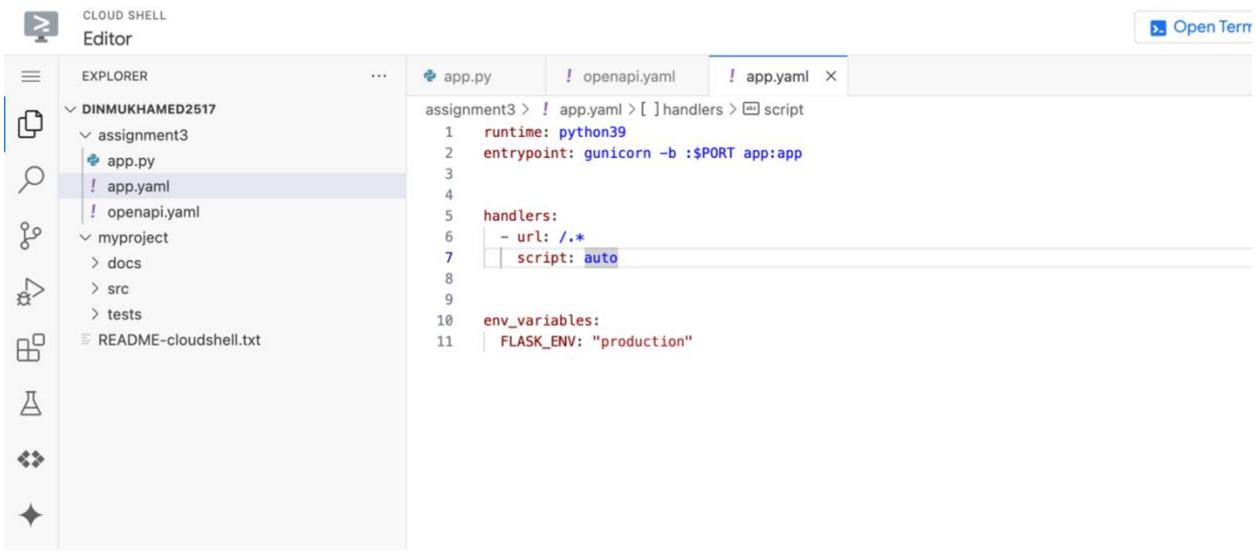
```yaml
openapi: 3.1.0
info:
  title: Hello World API
  description: A simple API to say hello
  version: 1.0.0
paths:
  /api/hello:
    get:
      summary: Returns a hello message
      responses:
        '200':
          description: A hello message
          content:
            application/json:
              schema:
                type: object
                properties:
                  message:
                    type: string
                    example: Hello, World!
```



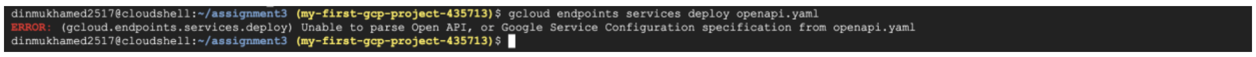
```python
from flask import Flask, jsonify

app = Flask(__name__)

@app.route('/api/hello', methods=['GET'])
def hello():
    return jsonify({'message': 'Hello, World!'})

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080, debug=True)
```



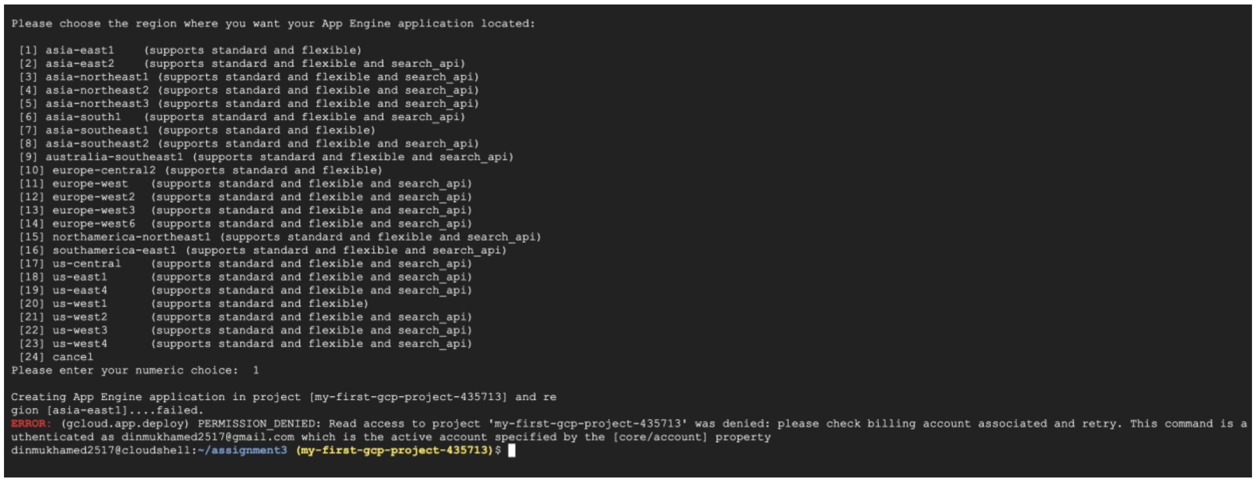
```
* Run `gcloud --help` to see the Cloud Platform services you can interact with. And run `gcloud help COMMAND` to get help on any gcloud c
* Run `gcloud topic --help` to learn about advanced features of the CLI like arg files and output formatting
* Run `gcloud cheat-sheet` to see a roster of go-to `gcloud` commands.
dinmukhamed2517@cloudshell:~ (my-first-gcp-project-435713)$ ls
assignment3  myproject  README-cloudshell.txt
dinmukhamed2517@cloudshell:~ (my-first-gcp-project-435713)$ cd assignment3/
dinmukhamed2517@cloudshell:~/assignment3 (my-first-gcp-project-435713)$ python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8080
 * Running on http://10.88.0.4:8080
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 216-205-559
127.0.0.1 - - [03/Nov/2024 11:59:50] "GET /?authuser=0&redirectedPreviously=true HTTP/1.1" 404 -
```



```yaml
runtime: python39
entrypoint: gunicorn -b :$PORT app:app


handlers:
  - url: /.*
    script: auto


env_variables:
  FLASK_ENV: "production"
```



```
dinmukhamed2517@cloudshell:~/assignment3 (my-first-gcp-project-435713)$ gcloud endpoints services deploy openapi.yaml
ERROR: (gcloud.endpoints.services.deploy) Unable to parse Open API, or Google Service Configuration specification from openapi.yaml
dinmukhamed2517@cloudshell:~/assignment3 (my-first-gcp-project-435713)$ 
```



```
Please choose the region where you want your App Engine application located:

 [1] asia-east1     (supports standard and flexible)
 [2] asia-east2     (supports standard and flexible and search_api)
 [3] asia-northeast1 (supports standard and flexible and search_api)
 [4] asia-northeast2 (supports standard and flexible and search_api)
 [5] asia-northeast3 (supports standard and flexible and search_api)
 [6] asia-south1    (supports standard and flexible and search_api)
 [7] asia-southeast1 (supports standard and flexible)
 [8] asia-southeast2 (supports standard and flexible and search_api)
 [9] australia-southeast1 (supports standard and flexible and search_api)
 [10] europe-central2 (supports standard and flexible)
 [11] europe-west    (supports standard and flexible and search_api)
 [12] europe-west2   (supports standard and flexible and search_api)
 [13] europe-west3   (supports standard and flexible and search_api)
 [14] europe-west6   (supports standard and flexible and search_api)
 [15] northamerica-northeast1 (supports standard and flexible and search_api)
 [16] southamerica-east1 (supports standard and flexible and search_api)
 [17] us-central     (supports standard and flexible and search_api)
 [18] us-east1       (supports standard and flexible and search_api)
 [19] us-east4       (supports standard and flexible and search_api)
 [20] us-west1       (supports standard and flexible)
 [21] us-west2       (supports standard and flexible and search_api)
 [22] us-west3       (supports standard and flexible and search_api)
 [23] us-west4       (supports standard and flexible and search_api)
 [24] cancel
Please enter your numeric choice:  1

Creating App Engine application in project [my-first-gcp-project-435713] and re
gion [asia-east1]....failed.
ERROR: (gcloud.app.deploy) PERMISSION_DENIED: Read access to project 'my-first-gcp-project-435713' was denied: please check billing account associated and retry. This command is a
uthenticated as dinmukhamed2517@gmail.com which is the active account specified by the [core/account] property
dinmukhamed2517@cloudshell:~/assignment3 (my-first-gcp-project-435713)$ 
```

Results

Successfully deployed the API, accessible via the provided URL. Verified the endpoint by calling the API and obtaining a response: { "message": "Hello, World!" }.

Exercise 2: Google Cloud Databases

Objective

To set up a Google Cloud SQL database and interact with it through a Python script.

Steps and Implementation

1. **Creating Cloud SQL Instance**:
   Set up a MySQL instance on Google Cloud SQL.
   Configured instance settings like region and machine type.
2. **Creating Database and Table**:
   Connected to the instance and created a database (sample_db) and a users table with sample data.

Code example:

```
CREATE DATABASE sample_db; USE sample_db; CREATE TABLE users ( id INT
AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100) NOT NULL, email
VARCHAR(100) NOT NULL ); INSERT INTO users (name, email) VALUES ('Alice',
'alice@example.com'); INSERT INTO users (name, email) VALUES ('Bob',
'bob@example.com');
```

```
import mysql.connector cnx = mysql.connector.connect( user='your-username', password='your-
password', host='your-cloud-sql-instance-ip', database='sample_db' ) cursor = cnx.cursor()
cursor.execute('SELECT * FROM users') for row in cursor: print(row) cursor.close() cnx.close()
```

Results

Successfully set up a Cloud SQL database with sample data.
Verified the connection by retrieving and displaying the data in the users table.

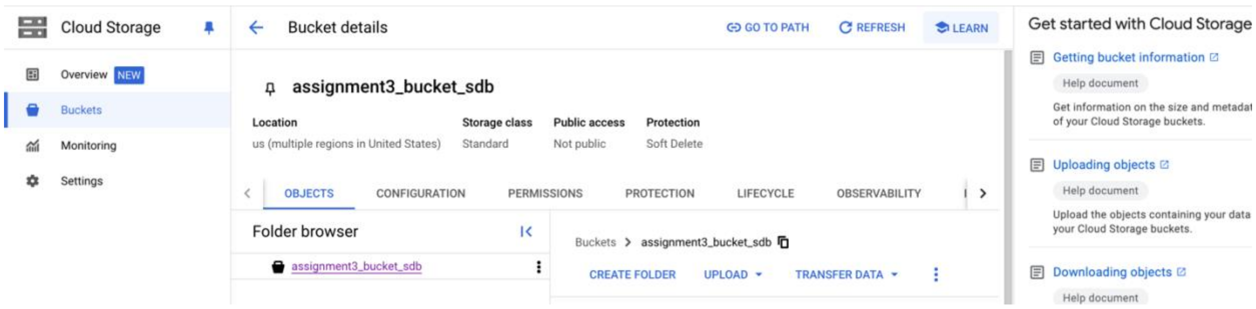
Exercise 3: Integrating Machine Learning with Google Cloud

Objective

To train and deploy a machine learning model using Google Cloud AI Platform.

Steps and Implementation

1. **Creating Cloud Storage Bucket**:
   Created a Cloud Storage bucket to store training data and model files.
2. **Preparing the Training Data**:
   Uploaded a dataset for classification to the Cloud Storage bucket.



Cloud Storage

Bucket details

GO TO PATH REFRESH LEARN

Overview NEW

Buckets

Monitoring

Settings

assignment3_bucket_sdb

Location	Storage class	Public access	Protection
us (multiple regions in United States)	Standard	Not public	Soft Delete

OBJECTS CONFIGURATION PERMISSIONS PROTECTION LIFECYCLE OBSERVABILITY

Folder browser

assignment3_bucket_sdb

Buckets > assignment3_bucket_sdb

CREATE FOLDER UPLOAD TRANSFER DATA

Get started with Cloud Storage

Getting bucket information

Help document

Get information on the size and metadat of your Cloud Storage buckets.

Uploading objects

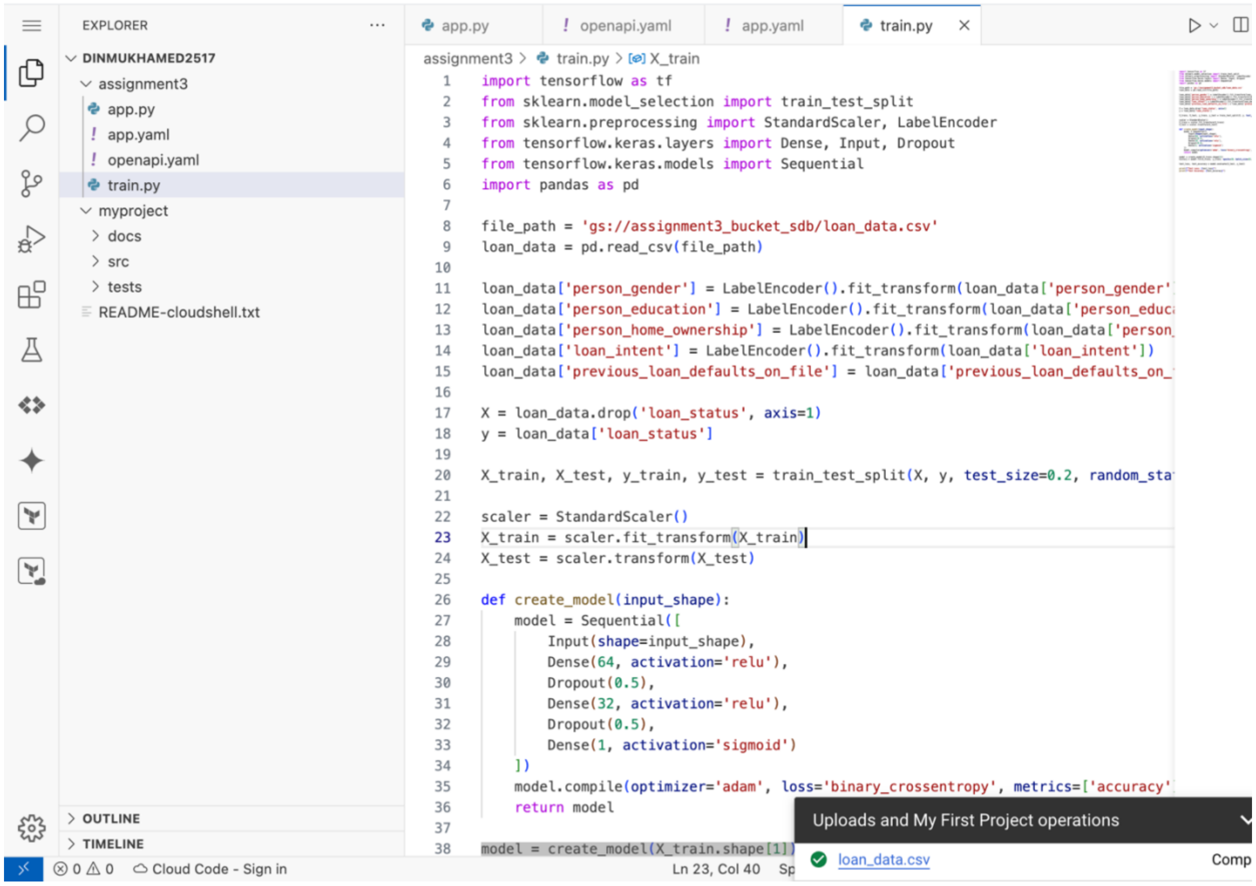
Help document

Upload the objects containing your data your Cloud Storage buckets.

Downloading objects

Help document

3. **Training Script**:



EXPLORER

DINMUKHAMED2517
- assignment3
  - app.py
  - app.yaml
  - openapi.yaml
  - train.py
- myproject
  - docs
  - src
  - tests
- README-cloudshell.txt

OUTLINE

TIMELINE

app.py | openapi.yaml | app.yaml | train.py

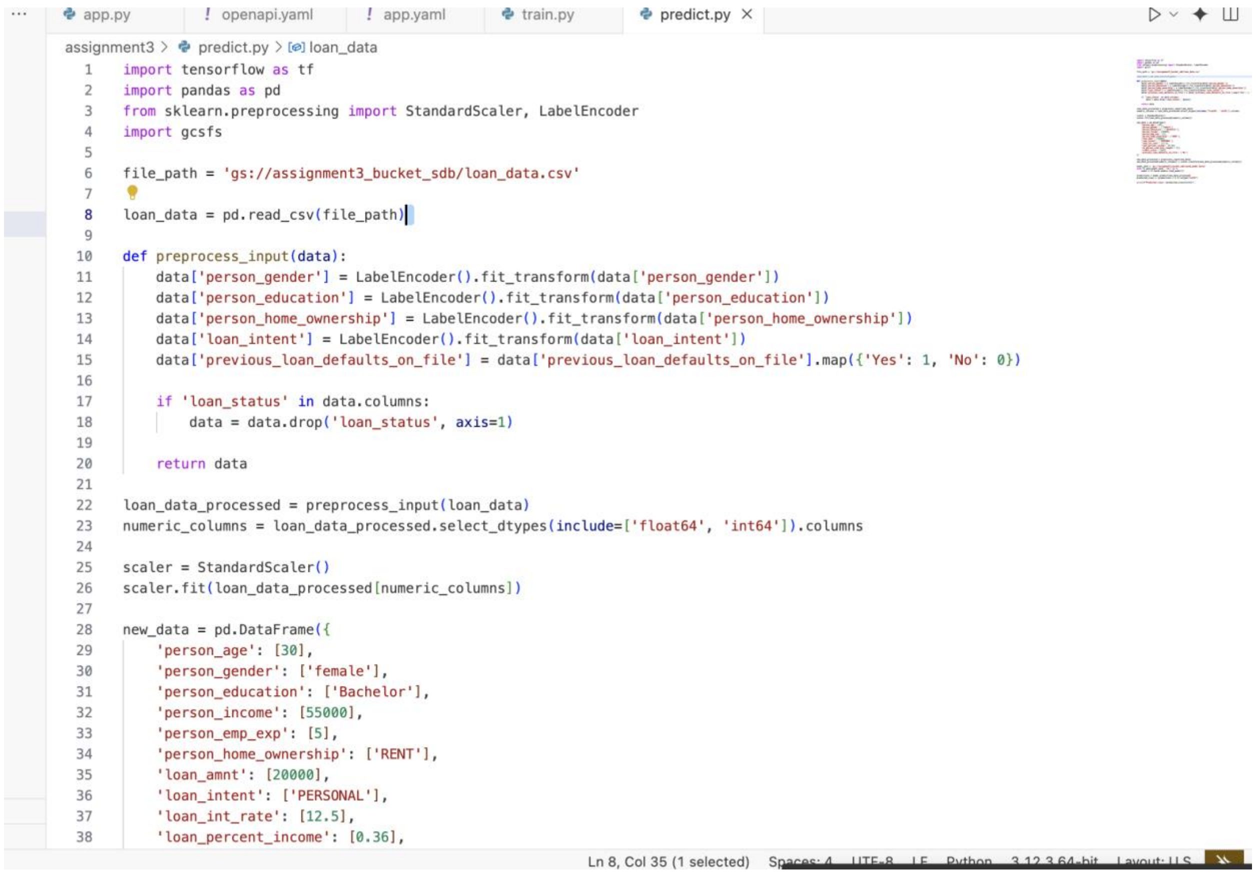
assignment3 > train.py > X_train

```python
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from tensorflow.keras.layers import Dense, Input, Dropout
from tensorflow.keras.models import Sequential
import pandas as pd

file_path = 'gs://assignment3_bucket_sdb/loan_data.csv'
loan_data = pd.read_csv(file_path)

loan_data['person_gender'] = LabelEncoder().fit_transform(loan_data['person_gender'
loan_data['person_education'] = LabelEncoder().fit_transform(loan_data['person_educa
loan_data['person_home_ownership'] = LabelEncoder().fit_transform(loan_data['person_
loan_data['loan_intent'] = LabelEncoder().fit_transform(loan_data['loan_intent'])
loan_data['previous_loan_defaults_on_file'] = loan_data['previous_loan_defaults_on_

X = loan_data.drop('loan_status', axis=1)
y = loan_data['loan_status']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

def create_model(input_shape):
    model = Sequential([
        Input(shape=input_shape),
        Dense(64, activation='relu'),
        Dropout(0.5),
        Dense(32, activation='relu'),
        Dropout(0.5),
        Dense(1, activation='sigmoid')
    ])
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'
    return model

model = create_model(X_train.shape[1])
```

Uploads and My First Project operations

loan_data.csv Comp

0 0 Cloud Code - Sign in Ln 23, Col 40 Sp

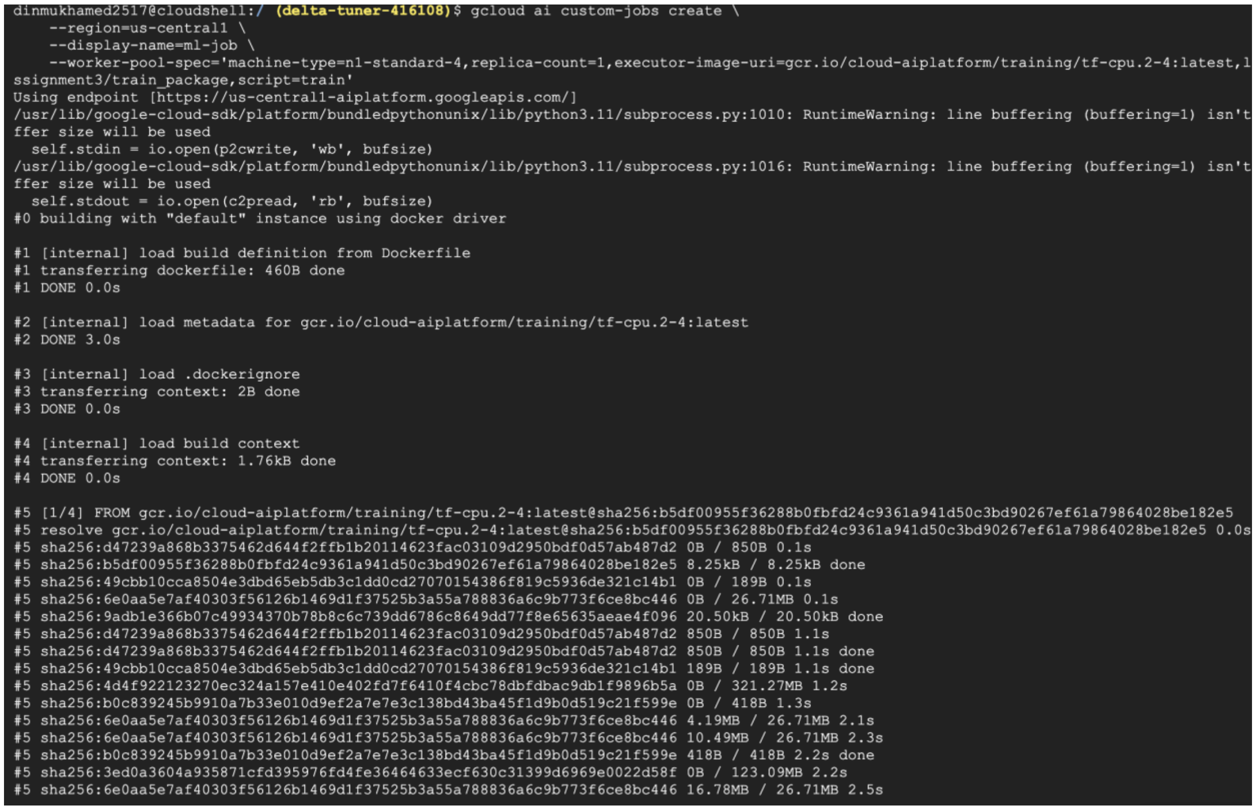


app.py | openapi.yaml | app.yaml | train.py | predict.py

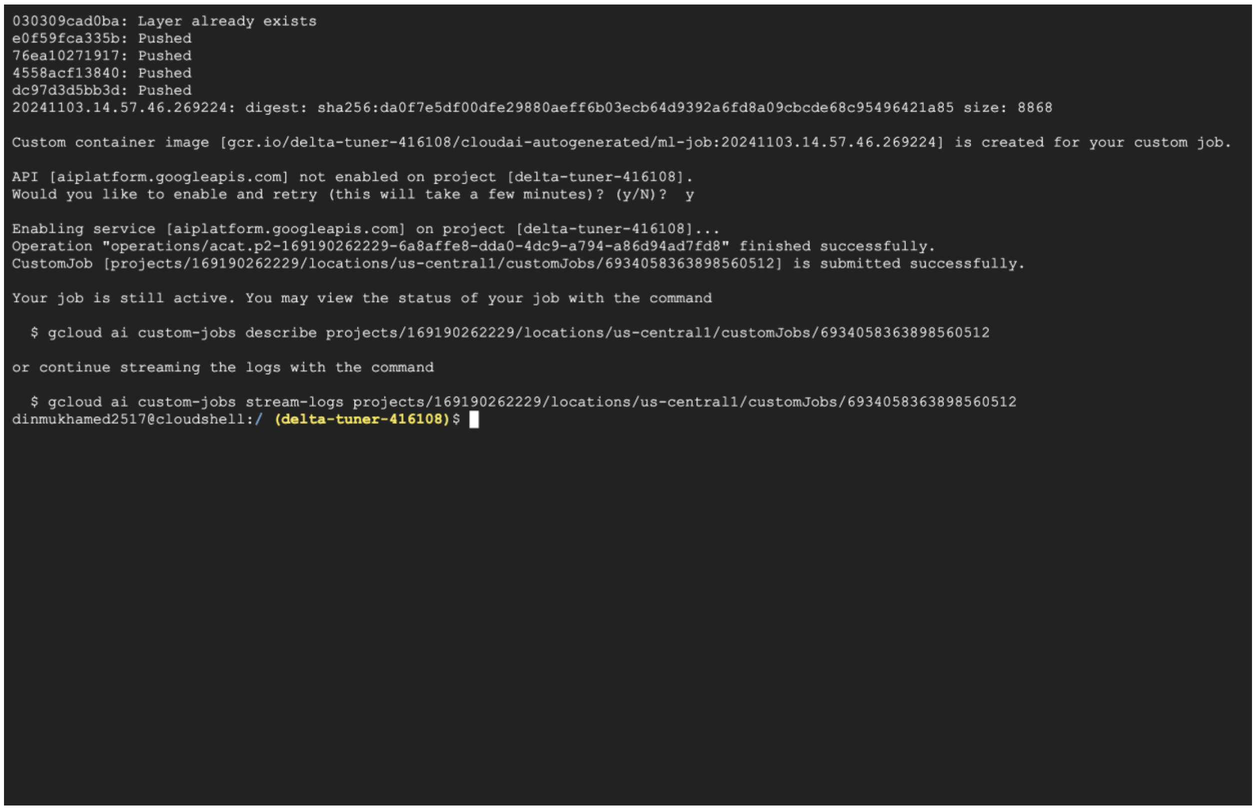
assignment3 > predict.py > loan_data

```python
import tensorflow as tf
import pandas as pd
from sklearn.preprocessing import StandardScaler, LabelEncoder
import gcsfs

file_path = 'gs://assignment3_bucket_sdb/loan_data.csv'

loan_data = pd.read_csv(file_path)

def preprocess_input(data):
    data['person_gender'] = LabelEncoder().fit_transform(data['person_gender'])
    data['person_education'] = LabelEncoder().fit_transform(data['person_education'])
    data['person_home_ownership'] = LabelEncoder().fit_transform(data['person_home_ownership'])
    data['loan_intent'] = LabelEncoder().fit_transform(data['loan_intent'])
    data['previous_loan_defaults_on_file'] = data['previous_loan_defaults_on_file'].map({'Yes': 1, 'No': 0})

    if 'loan_status' in data.columns:
        data = data.drop('loan_status', axis=1)

    return data

loan_data_processed = preprocess_input(loan_data)
numeric_columns = loan_data_processed.select_dtypes(include=['float64', 'int64']).columns

scaler = StandardScaler()
scaler.fit(loan_data_processed[numeric_columns])

new_data = pd.DataFrame({
    'person_age': [30],
    'person_gender': ['female'],
    'person_education': ['Bachelor'],
    'person_income': [55000],
    'person_emp_exp': [5],
    'person_home_ownership': ['RENT'],
    'loan_amnt': [20000],
    'loan_intent': ['PERSONAL'],
    'loan_int_rate': [12.5],
    'loan_percent_income': [0.36],
```

Ln 8, Col 35 (1 selected) Spaces: 4 UTF-8 LF Python 3.12.3 64-bit Layout: U.S.



```
dinmukhamed2517@cloudshell:/ (delta-tuner-416108)$ gcloud ai custom-jobs create \
    --region=us-central1 \
    --display-name=ml-job \
    --worker-pool-spec='machine-type=n1-standard-4,replica-count=1,executor-image-uri=gcr.io/cloud-aiplatform/training/tf-cpu.2-4:latest,l
ssignment3/train_package,script=train'
Using endpoint [https://us-central1-aiplatform.googleapis.com/]
/usr/lib/google-cloud-sdk/platform/bundledpythonunix/lib/python3.11/subprocess.py:1010: RuntimeWarning: line buffering (buffering=1) isn't
ffer size will be used
  self.stdin = io.open(p2cwrite, 'wb', bufsize)
/usr/lib/google-cloud-sdk/platform/bundledpythonunix/lib/python3.11/subprocess.py:1016: RuntimeWarning: line buffering (buffering=1) isn't
ffer size will be used
  self.stdout = io.open(c2pread, 'rb', bufsize)
#0 building with "default" instance using docker driver

#1 [internal] load build definition from Dockerfile
#1 transferring dockerfile: 460B done
#1 DONE 0.0s

#2 [internal] load metadata for gcr.io/cloud-aiplatform/training/tf-cpu.2-4:latest
#2 DONE 3.0s

#3 [internal] load .dockerignore
#3 transferring context: 2B done
#3 DONE 0.0s

#4 [internal] load build context
#4 transferring context: 1.76kB done
#4 DONE 0.0s

#5 [1/4] FROM gcr.io/cloud-aiplatform/training/tf-cpu.2-4:latest@sha256:b5df00955f36288b0fbfd24c9361a941d50c3bd90267ef61a79864028be182e5
#5 resolve gcr.io/cloud-aiplatform/training/tf-cpu.2-4:latest@sha256:b5df00955f36288b0fbfd24c9361a941d50c3bd90267ef61a79864028be182e5 0.0s
#5 sha256:d47239a868b3375462d644f2ffb1b20114623fac03109d2950bdf0d57ab487d2 0B / 850B 0.1s
#5 sha256:b5df00955f36288b0fbfd24c9361a941d50c3bd90267ef61a79864028be182e5 8.25kB / 8.25kB done
#5 sha256:49cbb10cca8504e3dbd65eb5db3c1dd0cd27070154386f819c5936de321c14b1 0B / 189B 0.1s
#5 sha256:6e0aa5e7af40303f56126b1469d1f37525b3a55a788836a6c9b773f6ce8bc446 0B / 26.71MB 0.1s
#5 sha256:9adb1e366b07c49934370b78b8c6c739dd6786c8649dd77f8e65635aeae4f096 20.50kB / 20.50kB done
#5 sha256:d47239a868b3375462d644f2ffb1b20114623fac03109d2950bdf0d57ab487d2 850B / 850B 1.1s
#5 sha256:d47239a868b3375462d644f2ffb1b20114623fac03109d2950bdf0d57ab487d2 850B / 850B 1.1s done
#5 sha256:49cbb10cca8504e3dbd65eb5db3c1dd0cd27070154386f819c5936de321c14b1 189B / 189B 1.1s done
#5 sha256:4d4f922123270ec324a157e410e402fd7f6410f4cbc78dbfdbac9db1f9896b5a 0B / 321.27MB 1.2s
#5 sha256:b0c839245b9910a7b33e010d9ef2a7e7e3c138bd43ba45f1d9b0d519c21f599e 0B / 418B 1.3s
#5 sha256:6e0aa5e7af40303f56126b1469d1f37525b3a55a788836a6c9b773f6ce8bc446 4.19MB / 26.71MB 2.1s
#5 sha256:6e0aa5e7af40303f56126b1469d1f37525b3a55a788836a6c9b773f6ce8bc446 10.49MB / 26.71MB 2.3s
#5 sha256:b0c839245b9910a7b33e010d9ef2a7e7e3c138bd43ba45f1d9b0d519c21f599e 418B / 418B 2.2s done
#5 sha256:3ed0a3604a935871cfd395976fd4fe36464633ecf630c31399d6969e0022d58f 0B / 123.09MB 2.2s
#5 sha256:6e0aa5e7af40303f56126b1469d1f37525b3a55a788836a6c9b773f6ce8bc446 16.78MB / 26.71MB 2.5s
```



```
030309cad0ba: Layer already exists
e0f59fca335b: Pushed
76ea10271917: Pushed
4558acf13840: Pushed
dc97d3d5bb3d: Pushed
20241103.14.57.46.269224: digest: sha256:da0f7e5df00dfe29880aeff6b03ecb64d9392a6fd8a09cbcde68c95496421a85 size: 8868

Custom container image [gcr.io/delta-tuner-416108/cloudai-autogenerated/ml-job:20241103.14.57.46.269224] is created for your custom job.

API [aiplatform.googleapis.com] not enabled on project [delta-tuner-416108].
Would you like to enable and retry (this will take a few minutes)? (y/N)?  y

Enabling service [aiplatform.googleapis.com] on project [delta-tuner-416108]...
Operation "operations/acat.p2-169190262229-6a8affe8-dda0-4dc9-a794-a86d94ad7fd8" finished successfully.
CustomJob [projects/169190262229/locations/us-central1/customJobs/6934058363898560512] is submitted successfully.

Your job is still active. You may view the status of your job with the command

  $ gcloud ai custom-jobs describe projects/169190262229/locations/us-central1/customJobs/6934058363898560512

or continue streaming the logs with the command

  $ gcloud ai custom-jobs stream-logs projects/169190262229/locations/us-central1/customJobs/6934058363898560512
dinmukhamed2517@cloudshell:/ (delta-tuner-416108)$ 
```

Results

Successfully trained, deployed, and tested the machine learning model.
Verified model predictions using the predict.py script.

Conclusion

In this assignment, we successfully explored key Google Cloud services, including Google Cloud Endpoints for API management, Cloud SQL for database management, and AI Platform for machine learning model deployment. These exercises provided a hands-on understanding of deploying and managing cloud applications and utilizing Google Cloud for robust cloud solutions.