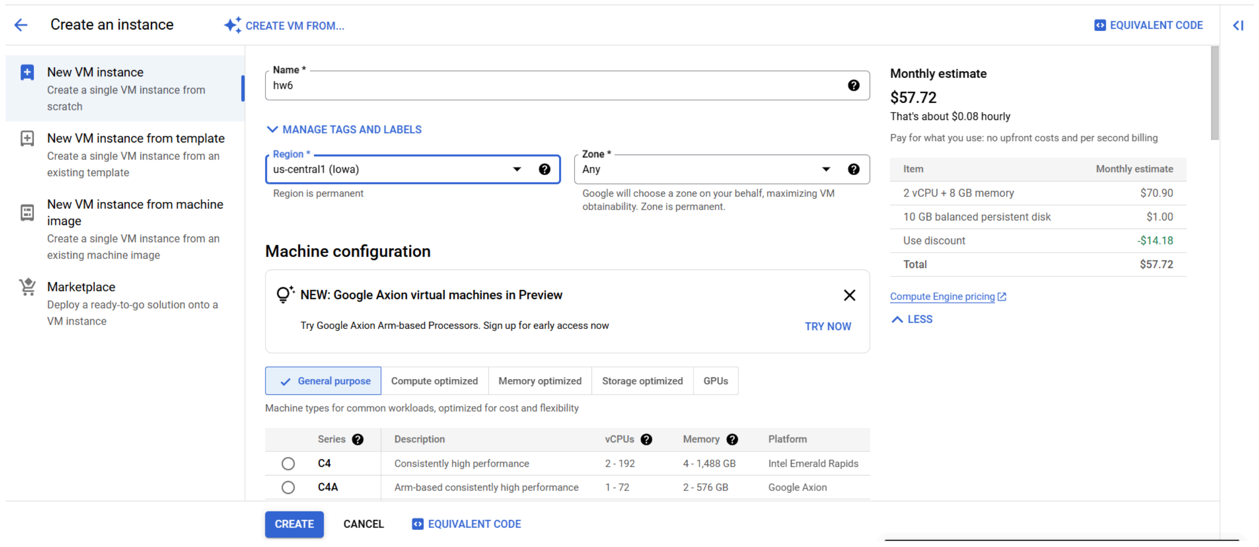
Name: Jeremy Bui
Course: DS 561

Homework 6

Setup

- Download hw5-db SQL, put into bucket, import it into my current SQL server I have from HW 5
- Create new vm for the app



Create an instance

Name *: hw6

Region *: us-central1 (Iowa)
Region is permanent

Zone *: Any
Google will choose a zone on your behalf, maximizing VM obtainability. Zone is permanent.

Machine configuration

NEW: Google Axion virtual machines in Preview
Try Google Axion Arm-based Processors. Sign up for early access now

General purpose | Compute optimized | Memory optimized | Storage optimized | GPUs

Machine types for common workloads, optimized for cost and flexibility

	Series	Description	vCPUs	Memory	Platform
○	C4	Consistently high performance	2 - 192	4 - 1,488 GB	Intel Emerald Rapids
○	C4A	Arm-based consistently high performance	1 - 72	2 - 576 GB	Google Axion

Monthly estimate
$57.72
That's about $0.08 hourly
Pay for what you use: no upfront costs and per second billing

Item	Monthly estimate
2 vCPU + 8 GB memory	$70.90
10 GB balanced persistent disk	$1.00
Use discount	-$14.18
Total	$57.72

CREATE VM FROM...
EQUIVALENT CODE
New VM instance
Create a single VM instance from scratch
New VM instance from template
Create a single VM instance from an existing template
New VM instance from machine image
Create a single VM instance from an existing machine image
Marketplace
Deploy a ready-to-go solution onto a VM instance
MANAGE TAGS AND LABELS
TRY NOW
Compute Engine pricing
LESS
CREATE
CANCEL
EQUIVALENT CODE

-
- gcloud compute instances set-service-account hw6 --zone=us-east1-c --service-account=844994497823-compute@developer.gserviceaccount.com --scopes=https://www.googleapis.com/auth/cloud-platformSsh (gcloud compute ssh webserver-vm --zone=us-central1-c, etc) into VMs and install dependencies
  - sudo apt-get install python3-venv
  - python3 -m venv venv
  - source venv/bin/activate
  - pip3 install google-cloud-storage google-cloud-logging mysql-connector-python SQLAlchemy scikit-learn pandas
- Add hw6's external IP to authorized networks

Code

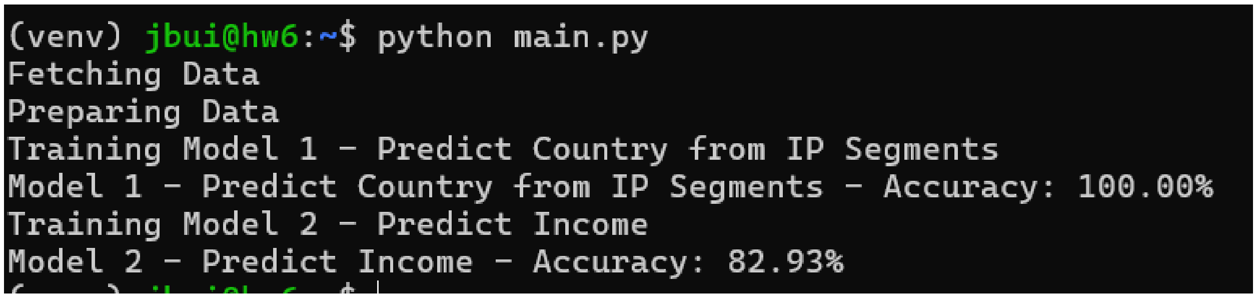
```
import pandas as pd
from sqlalchemy import create_engine
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline

DB_CONFIG = {
    "host": "34.42.100.68",
    "user": "root",
    "password": "",
    "database": "hw5-db"
}

def get_sqlalchemy_engine():
    return
create_engine(f"mysql+mysqlconnector://{DB_CONFIG['user']}:{DB_CONFIG['pas
sword']}@{DB_CONFIG['host']}/{DB_CONFIG['database']}")

# Fetch data from the database
def fetch_data():
    # Fetch relevant columns from the 'requests' table
    engine = get_sqlalchemy_engine()
    query = "SELECT client_ip, client_country, gender, age, is_banned,
income FROM requests"
    with engine.connect() as conn:
        data = pd.read_sql(query, conn)
    return data

# Split IP address into four separate segments
def split_ip(data):
    # Fill any missing 'client_ip' values with a default placeholder and
split into 4 segments
    data['client_ip'] = data['client_ip'].fillna('0.0.0.0')
    ip_split = data['client_ip'].str.split('.', expand=True).astype(int)
    ip_split.columns = ['ip_segment_1', 'ip_segment_2', 'ip_segment_3',
'ip_segment_4']
    data = data.join(ip_split).drop(columns=['client_ip'])
```

```python
    return data

# Prepare the data by encoding categorical features and splitting IP
segments
def prepare_data(data):
    data = split_ip(data)

    label_encoders = {
        "client_country": LabelEncoder(),
        "gender": LabelEncoder(),
        "age": LabelEncoder(),
        "income": LabelEncoder()
    }

    for column, encoder in label_encoders.items():
        data[column] = encoder.fit_transform(data[column])

    return data, label_encoders

# Train a RandomForest to predict 'client_country' from IP segments
def predict_country_from_ip(data):
    # Create an additional feature 'ip_sum' from the sum of IP segments
    data['ip_sum'] = data[['ip_segment_1', 'ip_segment_2', 'ip_segment_3',
'ip_segment_4']].sum(axis=1)
    X = data[['ip_segment_1', 'ip_segment_2', 'ip_segment_3',
'ip_segment_4', 'ip_sum']]
    y = data['client_country']

    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=561)

    model = RandomForestClassifier(n_estimators=500, max_depth=30,
min_samples_split=2, min_samples_leaf=1, class_weight='balanced',
random_state=561)
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Model 1 - Predict Country from IP Segments - Accuracy: 
{accuracy * 100:.2f}%")
    return model, accuracy

# Train a RandomForest to predict 'income' from other fields
def predict_income(data):
    data['ip_sum'] = data[['ip_segment_1', 'ip_segment_2', 'ip_segment_3', 
'ip_segment_4']].sum(axis=1)
    features = ['client_country', 'gender', 'age', 'is_banned', 'ip_sum']
    X = data[features]
    y = data['income']

    X_train, X_test, y_train, y_test = train_test_split(X, y, 
test_size=0.2, random_state=561)

    # Define a pipeline with scaling and RandomForestClassifier
    model = Pipeline([
        ('scaler', StandardScaler()),
        ('rf', RandomForestClassifier(n_estimators=500, max_depth=30, 
class_weight='balanced', random_state=561))
    ])
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)

    print(f"Model 2 - Predict Income - Accuracy: {accuracy * 100:.2f}%")
    return model, accuracy

def main():
    print("Fetching Data")
    data = fetch_data()

    print("Preparing Data")
    data, _ = prepare_data(data)

    print("Training Model 1 - Predict Country from IP Segments")
    model_country, country_accuracy = predict_country_from_ip(data)

    print("Training Model 2 - Predict Income")
    model_income, income_accuracy = predict_income(data)

if __name__ == "__main__":
    main()
```

- Setup connection to MySQL database using SQLAlchemy
- Fetch specific fields from the requests table to be used as features and target variables
- Use pd.read_sql to load the queried data into a pandas data frame
- Create a function split_ip
  - To handle missing data of the client_ip by filling them with a default value of ‘0.0.0.0’’
  - Split the ip into 4 segments
  - Covert them into an integer and make a new dataframe
  - Return this new dataframe with separated ip segments so this can be used as individual numerical features
- Create a prepare_data function
  - Call the split_ip function to add ip segment data
  - Initialize a dictionary of LabelEncoders to handle categorical fields like client_country, gender, age, income, etc. These will be converted to numerical codes
  - Iterate through each categorical column and apply the label encoding to ensure uniform data formatting across all features
  - Return back this dataframe with encoded values and dictionary of label encoders
- Function for model 1 of predict_country_from_ip:
  - Add the ip_sum to create a new feature for feature engineering
  - Create a X feature for the model which includes the ip segments and the ip_sum
  - Create a Y feature as the target variable of the client_country
  - Split the data into a training and testing set of 80-20 and a random state of 561
  - Create a RandomForestClassifer with the following parameters:
    - Set the number of trees in the forest to (n_estimators=500)
    - Limit the depth of the tree for overfitting (max_depth=30)
    - Set the min samples required before split (min_samples_split=2)
    - Set the class weight to balanced to avoid imbalances in weights for client country (class_weight=balanced)
    - Set random state

    - Train the model based on the training data of X_train and y_train to make predictions on the test of X_test.
    - Calculate the models accuracy by comparing predictions of y_pred with the actual values y_test.
- Function for model 2 to predict income based on the other features
    - Add the ip_sum to create a new feature for feature engineering
    - Create an X set of features that takes in the client_country, gender, age, is_banned, and ip_sum
    - Create a y to have the target variable of income
    - Split the data into training and testing with 80-20 split
    - Create a pipeline to have StandardScalar to standarize the features
    - Use RandomForestClassifier with same hyperparameters as before
    - Predict income on the test_data and calculate the accuracy by comparing y_pred with y_test

Result:

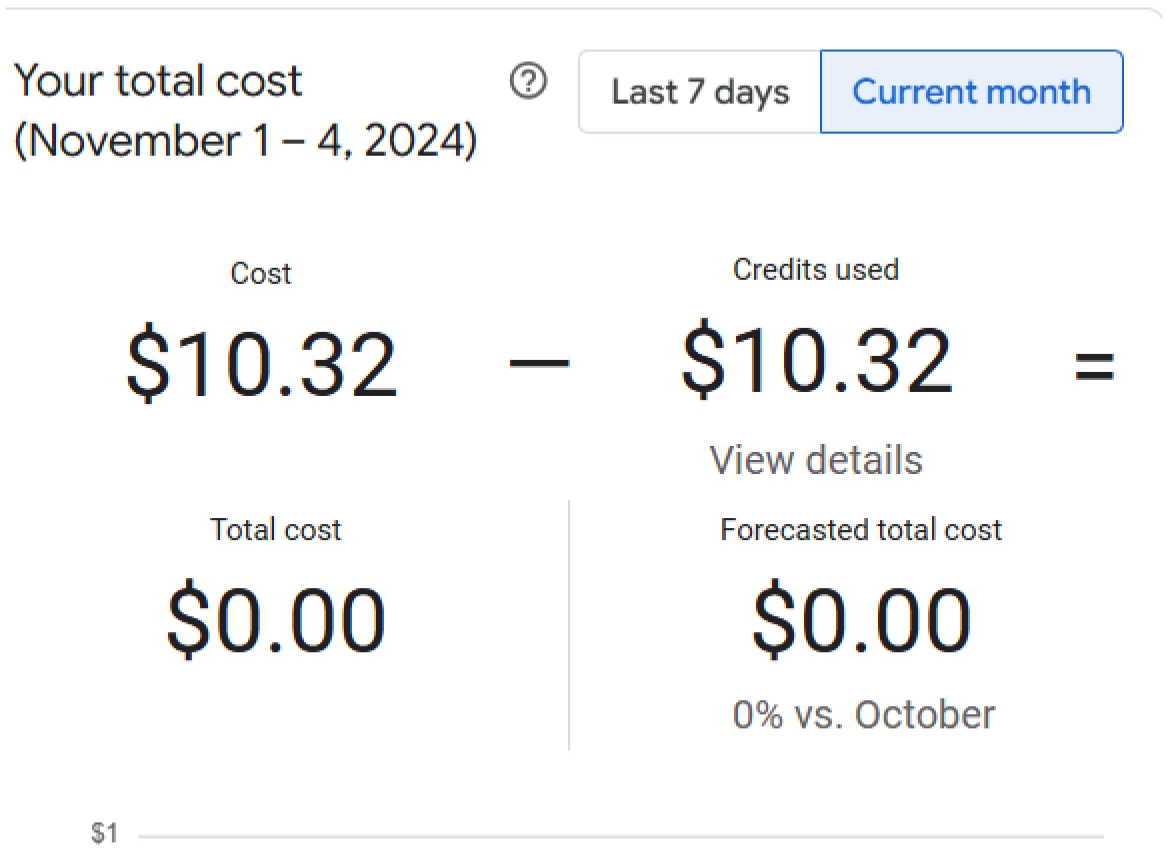


```
(venv) jbui@hw6:~$ python main.py
Fetching Data
Preparing Data
Training Model 1 - Predict Country from IP Segments
Model 1 - Predict Country from IP Segments - Accuracy: 100.00%
Training Model 2 - Predict Income
Model 2 - Predict Income - Accuracy: 82.93%
```

- Model Choice of Random Forest Classifier
    - The model works by creating multiple decision trees during training and outputting the classes of the individual tree.
    - It allows for reducing overfitting
    - Handles categorical and continuous data well
    - Allow us to tune parameters for the depth of the model
    - Performs well with larger datasets
- Feature Engineering
    - Split ip address into four segments to help the model understand patterns with each segment representing a distinct part of the ip structure
    - Sum the ips to provide a single measurement to capture the general magnitude of the ip address
    - Label encode to convert the categorical fields of client_country, gender, age, etc into numerical values

        - E.g client country will be transformed into a number where each number represents a different country
- Parameters of the model
    - Set the number of decision trees to 500
        - Initially, I had this at around 200 but was getting a poor accuracy score
    - Set the max depth of the tree to 30 to allow for detailed decisions but not overfit
        - This was initially at 20 but I was receiving a poor accuracy score as well
    - Set the minimum number of samples to split to 2
    - Create a class of balanced to ensure that the data with fewer samples also has influence in training
    - Set a pipeline for predict_income to standardize the feature
        - Allow for all features to be placed on a similar scale
        - Ensure the the scaling and model training is added for consistent performance
- Model Output
    - Model 1 predict the country from ip
        - I achieved a 100% accuracy which means that the model captured the geographical patterns from the ip data
        - This may be due to segmenting the ip address and adding the sum feature for the ip
    - Model 2 predict the income from the features
        - I achieved a 82.93% accuracy which shows the model can predict the income from demographic and ip features
        - This may be due ot the model using demographic data and ip to find patterns on the income levels. The prediction income has more complex factors which explains why it is lower than model 1’s accuracy

Cost:



Your total cost
(November 1 – 4, 2024)

Cost		Credits used	
$10.32	–	$10.32	=
		View details	

Total cost	Forecasted total cost
$0.00	$0.00
	0% vs. October