

# Empowering Mental Health Care with Azure: A Novel Approach to Predicting Depression Severity Using Machine Learning and the PHQ-9 Questionnaire

**Author:** **Ramakrishnan V**, *Microsoft Certified Trainer, Solutions Architect (7x Microsoft Azure Certified Professional)*

## Source code

<https://github.com/rama1azure/mental-health-prediction.git>

## Mental Health prediction application

[https://red-mushroom-071a67d10.2.azurestaticapps.net/login-with\\_admin\\_credentials.html](https://red-mushroom-071a67d10.2.azurestaticapps.net/login-with_admin_credentials.html)

## Introduction

Depression is a widespread mental health issue affecting millions of individuals worldwide. Early detection and accurate severity assessment can significantly improve the effectiveness of intervention and treatment strategies. The PHQ-9 (Patient Health Questionnaire-9) is a widely-used self-reporting tool for evaluating the severity of depression. This project aims to leverage the power of Azure services which includes AutoML to develop a predictive model based on the PHQ-9 questionnaire data, Azure Storage account, Azure App Service, Static Web app facilitating timely and efficient severity assessment of depression. We will also discuss the challenges and benefits of our solution, and provide insights into the next steps for improving our model and its deployment.

## Problem Statement

Our goal is to develop an accurate machine learning model that can predict the severity of depression for a given individual based on a set of input features from PHQ-9 questionnaire. We want to deploy this model as an API endpoint that can be consumed by other applications and services, while ensuring that the deployment is secure, scalable, and maintainable.

## Solution

To achieve our goal, we follow these steps:

1. Prepare the dataset and train a ML model - Classifier using Azure AutoML
2. Create a Flask application to serve the model predictions as an API
3. Deploy the Flask application to a Virtual Machine (VM)
4. Test the API and gather feedback from end-users

5. Containerize the Flask application using Docker
6. Push the Docker image to Azure Container Registry (ACR)
7. Deploy the Docker container as an Azure Web App
8. Build a web application to gather feedback and call the API for depression

severity predictions

9. Deploy the web application as a static app

## About Dataset used in this application

The **Patient Health Questionnaire-9 (PHQ-9)** is a widely used self-administered diagnostic instrument for assessing the severity of depression. Developed by Drs. Robert L. Spitzer, Janet B.W. Williams, Kurt Kroenke, and colleagues, the PHQ-9 is based on the criteria for diagnosing depressive disorders outlined in the Diagnostic and Statistical Manual of Mental Disorders, Fourth Edition (DSM-IV).

The PHQ-9 consists of nine items that correspond to the nine symptoms of major depressive disorder as defined by the DSM-IV. Each item is scored on a scale from 0 to 3, with the following response options:

- *0: Not at all*
- *1: Several days*
- *2: More than half the days*
- *3: Nearly every day*

The items on the PHQ-9 questionnaire are:

- 1. Little interest or pleasure in doing things*
- 2. Feeling down, depressed, or hopeless*
- 3. Trouble falling or staying asleep, or sleeping too much*
- 4. Feeling tired or having little energy*
- 5. Poor appetite or overeating*
- 6. Feeling bad about yourself or that you are a failure or have let yourself or your family*

*down*

- 7. Trouble concentrating on things, such as reading the newspaper or watching television*
- 8. Moving or speaking so slowly that other people could have noticed, or the opposite - being so fidgety or restless that you have been moving around a lot more than usual*
- 9. Thoughts that you would be better off dead or of hurting yourself in some way*

The total score on the PHQ-9 ranges from 0 to 27, with higher scores indicating more severe depression. The scores can be interpreted as follows:

- *0-4: Minimal depression*
- *5-9: Mild depression*
- *10-14: Moderate depression*
- *15-19: Moderately severe depression*
- *20-27: Severe depression*

The PHQ-9 is a valuable tool for clinicians to screen for depression, monitor the severity of the condition, and track the patient's response to treatment. In the context of our machine learning model, we can use the PHQ-9 questionnaire as a set of input features to predict depression severity, which can help guide appropriate interventions and support.

The scores from these 9 questions can be used to quantify the severity of depression and determine a treatment plan.

Here is a sample dataset based on the 9 questions from the Patient Health Questionnaire (PHQ-9):

**ID, Age, Gender, Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, PHQ9\_Total, Depression\_Severity**

1,23,F,0,1,0,0,1,0,1,1,0,4,Minimal  
2,34,M,2,2,1,1,2,2,3,3,1,17,Severe  
3,45,F,1,0,1,0,0,0,0,0,0,2,Minimal  
4,56,M,2,1,3,2,3,3,3,1,1,19,Severe  
5,29,F,1,1,1,0,0,0,1,1,0,5,Mild

This sample dataset consists of 5 patients, each with a score for each of the 9 questions. The severity of depression is determined by summing the scores for each question and using a cutoff score to categorize the severity as mild, moderate, or severe.

This sample dataset has been extrapolated and used to train an AI model to predict depression based on answers to the 9 questions. The AI model could use the scores for each question as input features, and the severity of depression as the target variable. The goal of the AI model would be to predict the severity of depression based on answers to the 9 questions.

## Technical Solution

We propose a machine learning-based solution using Azure AutoML to predict depression severity from PHQ-9 questionnaire responses.

To implement this solution, start by collecting and preprocessing the PHQ-9 questionnaire data. Next, use Azure AutoML for model training and evaluation. Deploy the best-performing model using Azure Web App Service, create an API endpoint for predictions, and develop a frontend for user interaction. By following these steps, you can create a robust and accurate system for predicting depression severity using the PHQ-9 questionnaire and Azure AI.

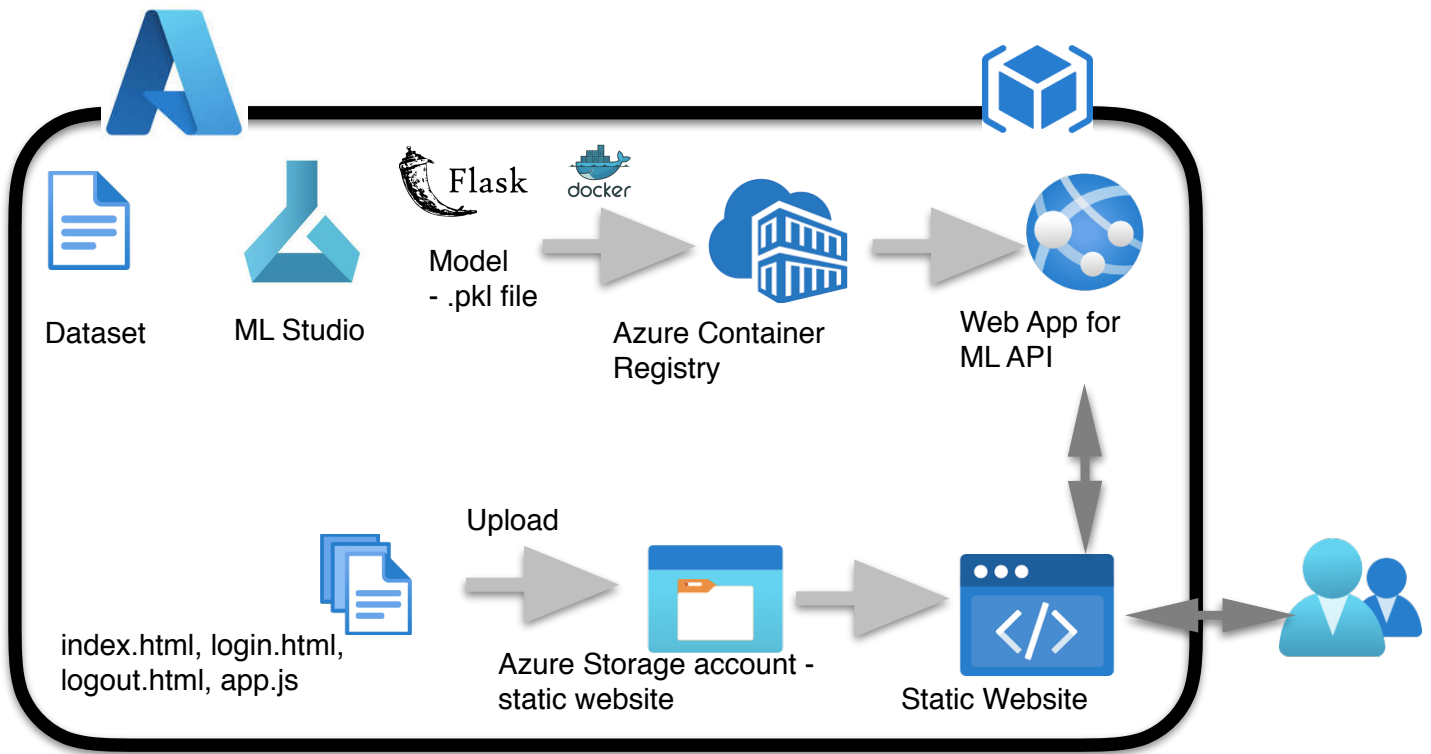
The solution involves the following steps:

1. **Dataset:** We use a dataset containing various features related to individuals' mental health, such as age, gender, and self-reported symptoms.
2. **Azure AutoML:** We utilize Azure AutoML to automatically train and select the Best Classifier on our dataset.
3. **Flask Application:** We create a Flask application to serve the trained model as an API endpoint.
4. **Virtual Machine:** We deploy our Flask application to a VM for testing and gathering feedback.
5. **Docker:** We containerize our Flask application using Docker to create a portable and scalable deployment.
6. **Azure Container Registry:** We push the Docker image to ACR for easy deployment and management.
7. **Azure Web App:** We deploy the Docker container as an Azure Web App, which serves as our ML API endpoint.
8. **Web Application:** We build a web application for users to submit their feedback and receive depression severity predictions.
9. **Static App:** We deploy the web application as a static app for easy access and scalability.

# All services used for this project in Azure

Name	Type	Location	Resource Group	Subscription	Last accessed
mentalhealthredirect	Static Web App	Central US	azure.blogathan	Visual Studio Enterprise Subscription	16 minutes ago
mental-health	Virtual machine	Central US	azure.blogathan	Visual Studio Enterprise Subscription	21 minutes ago
mentalhealthredirect	Storage account	Central US	azure.blogathan	Visual Studio Enterprise Subscription	24 minutes ago
azure.blogathan	Resource group	Central US	azure.blogathan	Visual Studio Enterprise Subscription	37 minutes ago
mentalhealthseventyprediction	App Service	East US	azure.blogathan	Visual Studio Enterprise Subscription	48 minutes ago
mentalhealthredirect	App Service	East US	azure.blogathan	Visual Studio Enterprise Subscription	4 hours ago
mentalhealthredirect	Container registry	Central US	azure.blogathan	Visual Studio Enterprise Subscription	4 hours ago
mental_health	Azure Machine Learning workspace	Central US	azure.blogathan	Visual Studio Enterprise Subscription	6 hours ago
ASP-azureblogathan-9763	App Service plan	East US	azure.blogathan	Visual Studio Enterprise Subscription	15 hours ago
mentalhealthredirect	Function App	East US	azure.blogathan	Visual Studio Enterprise Subscription	16 hours ago
mental-health-ip	Public IP address	Central US	azure.blogathan	Visual Studio Enterprise Subscription	17 hours ago
Visual Studio Enterprise Subscription	Subscription	Central US	azure.blogathan	Visual Studio Enterprise Subscription	6 months ago

## Technical Architecture Diagram of this Project



## Dataset and Training Process with Auto ML and Random Forest Classifier

We use a dataset containing various features related to individuals' mental health, such as age, gender, and self-reported symptoms. We preprocess the dataset by performing feature engineering and data normalization to ensure the best possible model performance. We then split the dataset into a training and testing set, with 80% of the data used for training and 20% for testing.

Next, we utilize Azure AutoML to automatically train a Classifier on our dataset. Azure AutoML iteratively trains and evaluates multiple models using different algorithms and hyperparameters, selecting the best performing model based on cross-validation results. After training, we save the best model as a **.pkl** file for deployment.

Microsoft Azure Machine Learning Studio

Default Directory > mental\_health > Automated ML

### Automated ML

Let Automated ML train and find the best model based on your data without writing a single line of code. [Learn more about Automated ML](#)

+ New Automated ML job Refresh

#### Recent Automated ML jobs

Display name	☆ Experiment	Status	Created on	Duration	Created by	Compute target	Tags
joyful_battery_q4dggpy2	mental_health_pr...	Com	Apr 1, 2023 6:38 PM	45m 12s	Ramakrishnan ...	mental-health-...	dynamic_allowlisti ...
coral_island_pt3m332r	mental_health_pr...	Com	Apr 1, 2023 6:37 PM	49m 45s	Ramakrishnan ...	mental-health-...	dynamic_allowlisti ...

#### Documentation

- Concept: What is Automated ML?
- Tutorial: Create your first classification model with Automated ML
- Blog: Build more accurate forecasts with new capabilities in Automated ML

Using Azure Automated ML service, upload the dataset and start the training process. Upon completion of training process, we can review the detail of trained model, test results as shown in below images:

**Best model: MaxAbsScaler, LightGBM with AUC weighted: 1 is selected by Automated ML**

Microsoft Azure Machine Learning Studio

Default Directory > mental\_health > Automated ML > mental\_health\_prediction > coral\_island\_pt3m332r

coral\_island\_pt3m332r ✎ ☆ ✔ Completed

Overview Data guardrails Models Outputs + logs Child jobs

Refresh Edit and submit (preview) Register model Cancel Delete Compare (preview)

**Properties**

Status ✔ Completed

**Warning:** No scores improved over last 20 iterations, so experiment stopped early. This early stopping behavior can be disabled by setting enable\_early\_stopping = False in AutoMLConfig for notebook/python SDK runs.

[See more details](#)

Created on Apr 1, 2023 6:37 PM

Start time Apr 1, 2023 6:37 PM

Duration 49m 44.70s

Compute duration 49m 44.70s

Compute target [mental-health-predict](#)

Name AutoML\_4182bb71-4cc4-4069-bb50-fd77d23cdd7b

**Inputs**

Input name: training\_data  
Dataset: [mental\\_health\\_prediction:1](#)

**Outputs**

Output name: best\_model  
Model: [azureml\\_AutoML\\_4182bb71-4cc4-4069-bb50-fd77d23cdd7b\\_0\\_output\\_mlflow\\_log\\_model\\_1299562991:1](#)

**Best model summary**

Algorithm name  
[MaxAbsScaler, LightGBM](#)

Hyperparameters  
[View hyperparameters](#)

AUC weighted  
1.00000 [View all other metrics](#)

Sampling  
100.00 %

Registered models

Microsoft Azure Machine Learning Studio

Default Directory

Jobs > mental\_health\_prediction > joyful\_battery\_q4dggpy2 > elated\_kiwi\_xrx4sg5n > sweet\_corn\_7jlg1zfl

sweet\_corn\_7jlg1zfl ✎ ☆ ✔ Completed

Overview Metrics Images Child jobs Outputs + logs Code Monitoring

Name  
1eb4dc29-ce46-48a3-a10f-63313284a82d

Script name  
model\_test.py

Created by  
Ramakrishnan Vedanarayanan

Job type  
Model test

Experiment  
[mental\\_health\\_prediction](#)

Environment  
[AzureML-AutoML:134](#)

Arguments  
None

Registered models  
None

See all properties

[Raw JSON](#)

Compute

**Run Metrics**

AUC\_macro  
1

AUC\_micro  
1

AUC\_weighted  
1

accuracy  
1

accuracy\_table  
azureml.v2.accuracy\_table

average\_precision\_score\_macro  
1

average\_precision\_score\_micro  
1

average\_precision\_score\_weighted  
1

balanced\_accuracy  
1

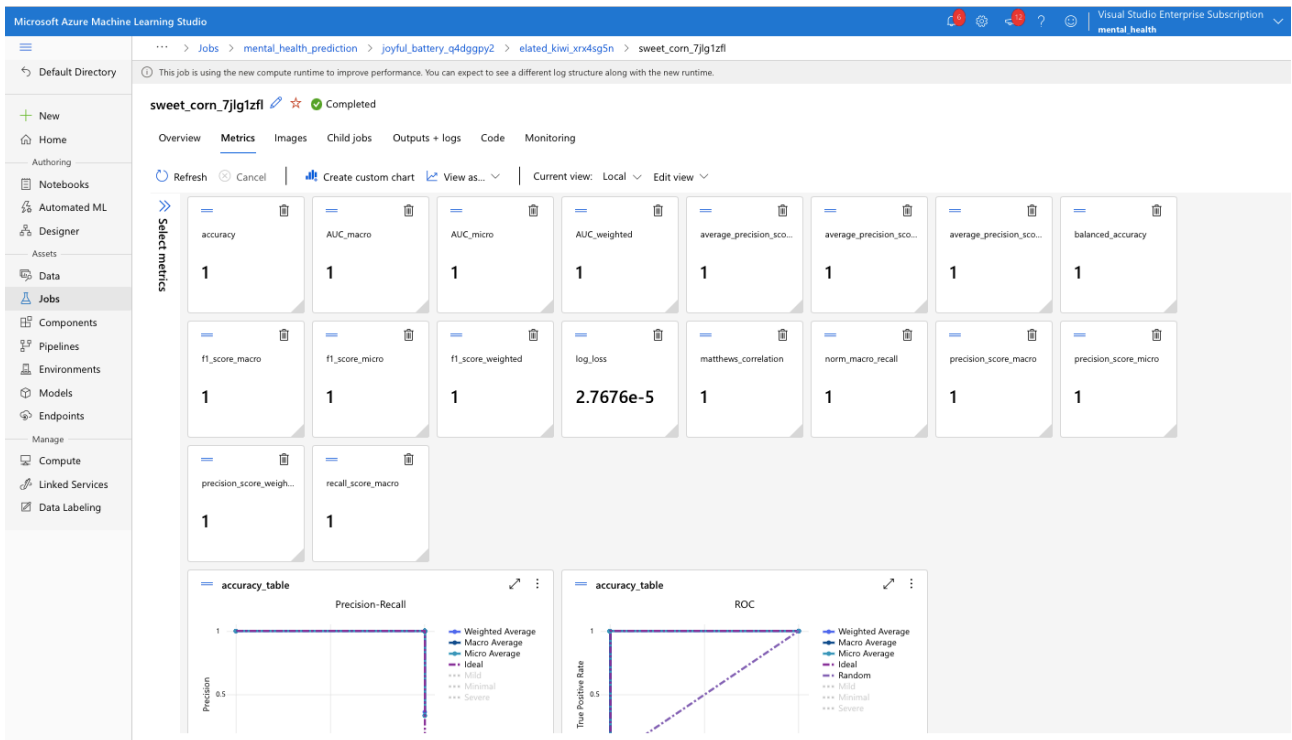
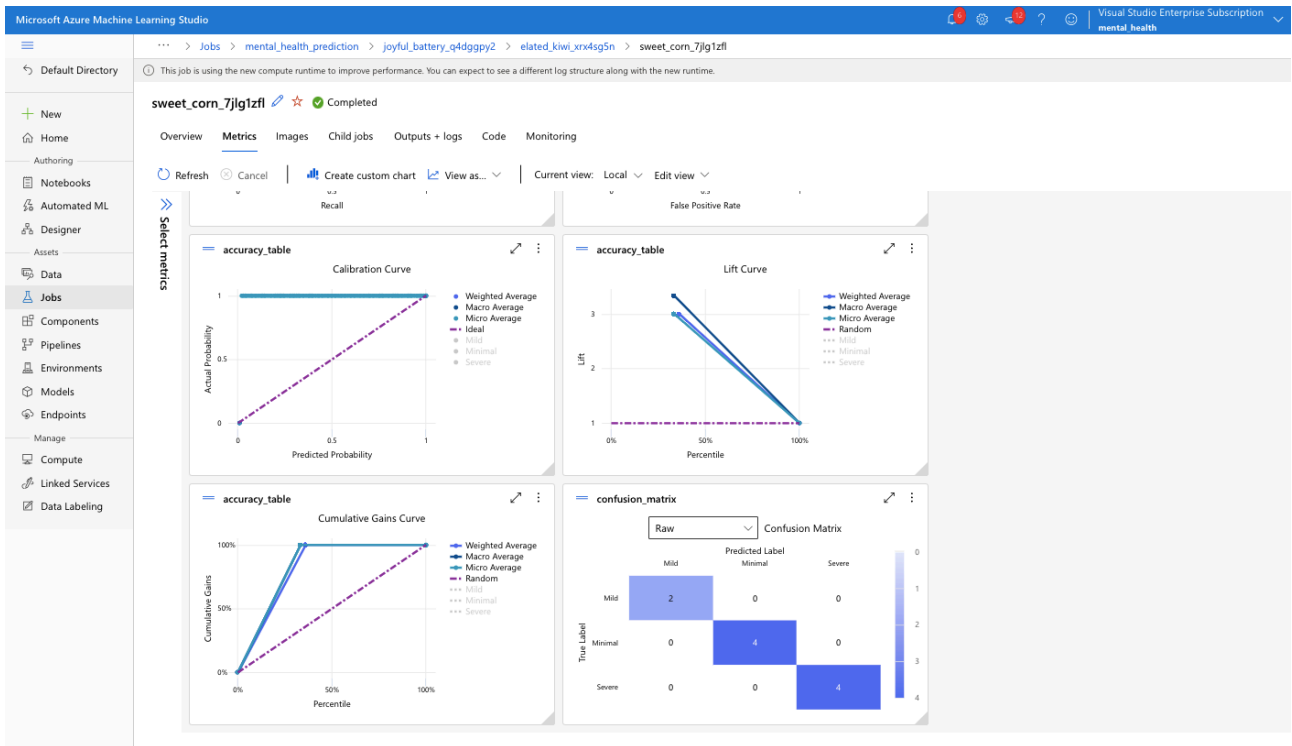
confusion\_matrix  
azureml.v2.confusion\_matrix

f1\_score\_macro  
1

f1\_score\_micro  
1

f1\_score\_weighted  
1

Additional metrics to review the accuracy of ML model selected by Automated ML  
Accuracy and other ML related key metrics of the Best model is shown below:



ML Model generated by Automated ML is shown below:

Microsoft Azure Machine Learning Studio

Default Directory > mental\_health > Models > azureml\_AutoML\_4182bb71-4cc4-4069-bb50-fd77d23cdd7b\_0\_output\_mlflow\_log\_model\_1299562991:1

Default Directory

azureml\_AutoML\_4182bb71-4cc4-4069-bb50-fd77d23cdd7b\_0\_output\_mlflow\_log\_model\_1299562991:1

Details Artifacts Jobs Data Responsible AI Explanations (preview) Fairness (preview)

Refresh Download all

mlflow-model

- conda.yaml
- MLmodel
- model.pkl
- python\_env.yaml
- requirements.txt

File Explorer Pane

## As an alternate approach of preparing ML model, used Random forest classifier to generate ML model (.pkl) files

```

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import accuracy_score, classification_report
from sklearn.ensemble import RandomForestClassifier
import pickle

data = pd.read_csv('depression_dataset_with_severity.csv')

# Preprocess the data
data['Gender'] = LabelEncoder().fit_transform(data['Gender'])
X = data[['Age', 'Gender', 'Q1', 'Q2', 'Q3', 'Q4', 'Q5', 'Q6', 'Q7', 'Q8', 'Q9']]
y = data['Depression_Severity']

# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scale the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Build the RandomForest model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Make predictions and evaluate the model
y_pred = model.predict(X_test)
print('Accuracy:', accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))

# Save the model and scaler as .pkl files
with open('model.pkl', 'wb') as f:
    pickle.dump(model, f)

with open('scaler.pkl', 'wb') as f:
    pickle.dump(scaler, f)

Accuracy: 1.0
      precision    recall  f1-score   support

    Mild         1.00      1.00      1.00         3
   Minimal         1.00      1.00      1.00        10
    Severe         1.00      1.00      1.00         7

 accuracy
macro avg         1.00      1.00      1.00        20
weighted avg         1.00      1.00      1.00        20

```

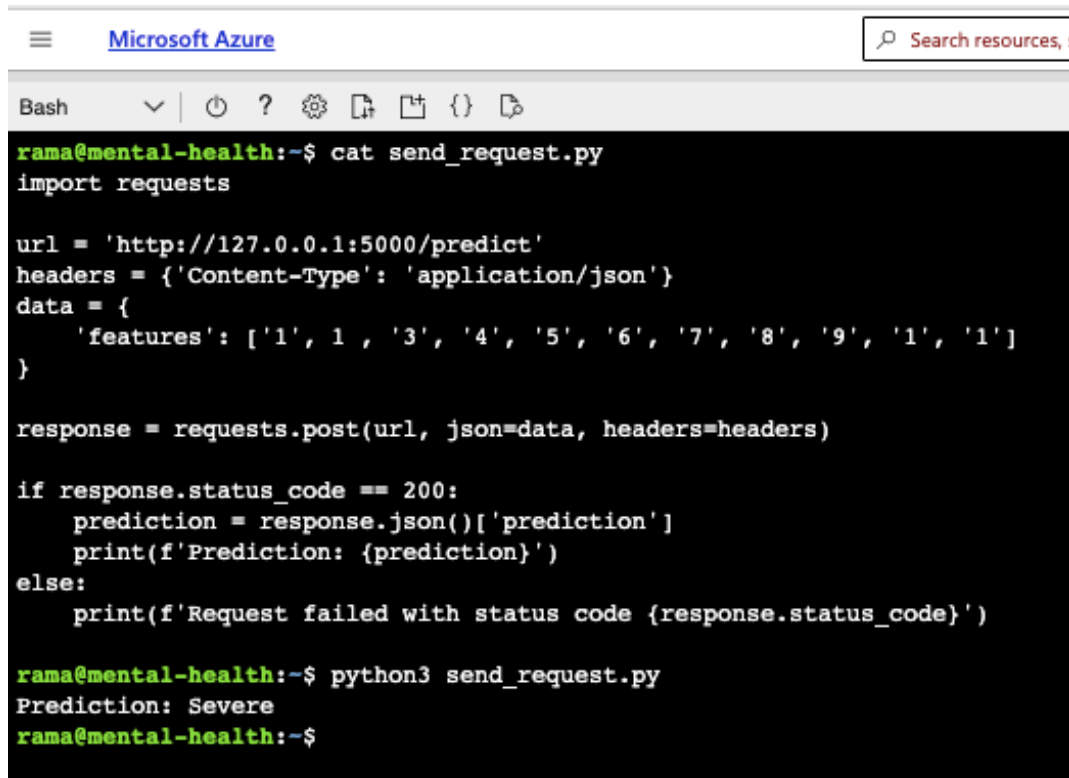


## Flask Application and Deployment to VM

We create a Flask application that loads the saved `.pkl` file and exposes an API endpoint for making depression severity predictions. The API accepts a POST request containing the input features in JSON format, and it returns the predicted severity as a JSON response.

We configure the VM (with networking Inbound rules to open the port) to run the Flask application with the necessary dependencies and expose the API endpoint to the internet.

As shown below, we get the prediction result for the test data sent as POST request to Flask API endpoint:



```
Microsoft Azure
Search resources, s

Bash
rama@mental-health:~$ cat send_request.py
import requests

url = 'http://127.0.0.1:5000/predict'
headers = {'Content-Type': 'application/json'}
data = {
    'features': ['1', '1', '3', '4', '5', '6', '7', '8', '9', '1', '1']
}

response = requests.post(url, json=data, headers=headers)

if response.status_code == 200:
    prediction = response.json()['prediction']
    print(f'Prediction: {prediction}')
else:
    print(f'Request failed with status code {response.status_code}')

rama@mental-health:~$ python3 send_request.py
Prediction: Severe
rama@mental-health:~$
```

## Testing and Gathering Feedback

We test the API endpoint by sending sample requests and verifying the correctness of the predictions. We also share the API with a group of end-users, who provide feedback on the model's performance and the overall user experience. This feedback helps us identify potential issues and areas for improvement.

## Containerizing with Docker and Pushing to ACR

To make our deployment more portable, scalable, and maintainable, we containerize the Flask application using Docker. We create a Dockerfile that describes the necessary dependencies, the application code, and how the application should be executed. We then build a Docker image using the Dockerfile.

Next, we push the Docker image to Azure Container Registry (ACR), a private registry for storing and managing Docker images. This allows us to easily deploy and manage our application in Azure.

## Dockerfile

```
# Use an official Python runtime as a parent image
FROM python:3.8-slim

# Set the working directory
WORKDIR /app

# Copy the requirements file into the container
COPY requirements.txt /app

# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Copy the rest of the application code into the container
COPY . /app

# Make port 80 available to the world outside this container
EXPOSE 80

# Define environment variable
ENV FLASK_APP=app.py

# Run the command to start the Flask app
CMD ["flask", "run", "--host=0.0.0.0", "--port=80"]
```

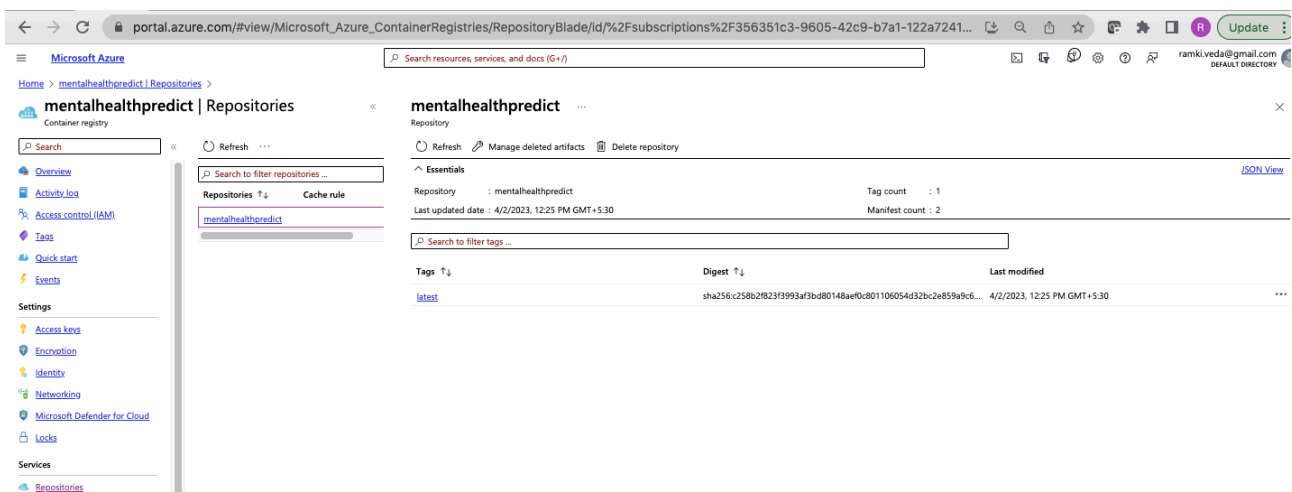
Ensure to place the Dockerfile in the same directory where .pkl files and app.py (flask program) are located

To get a Docker image, execute this command:  
`docker build -t my-ml-model .`

Post creating the docker image, tag using the below command:  
`sudo docker tag mental-health-predict:latest mentalhealthpredict.azurecr.io/mentalhealthpredict:latest`

Using, `sudo docker push`, push the image to azure container registry, Ensure to authenticate to ACR before pushing the image.

Post Image get pushed to Azure Container Registry, will be shown as below:

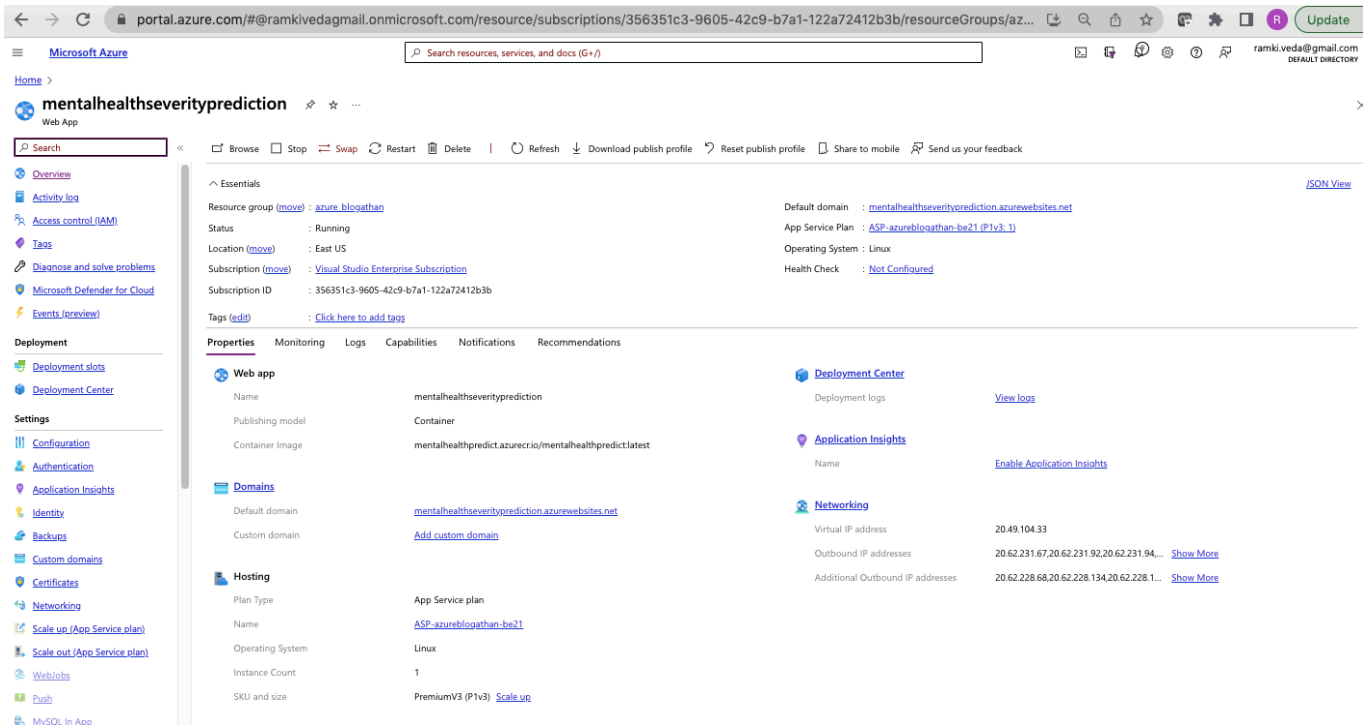


The screenshot displays the Azure Portal interface for the 'mentalhealthpredict' repository. The left sidebar shows navigation options like Overview, Activity log, and Settings. The main content area shows the repository details, including the repository name 'mentalhealthpredict', the last updated date '4/2/2023, 12:25 PM GMT+5:30', and the manifest count '2'. A table lists the tags, with the 'latest' tag having a digest of 'sha256:c258b2823f3993af3bd80148ae0c801106054d32bc2e859a9c6...' and a last modified date of '4/2/2023, 12:25 PM GMT+5:30'.

Tags	Digest	Last modified
latest	sha256:c258b2823f3993af3bd80148ae0c801106054d32bc2e859a9c6...	4/2/2023, 12:25 PM GMT+5:30

## Deploying as Azure Web App

We deploy the Docker container as an Azure Web App, which serves as our ML API endpoint. Azure Web Apps provide a fully managed platform for running containerized applications, which simplifies deployment and scaling. We configure the Web App to pull the Docker image from ACR and run it with the required resources.



## Building the Web Application

We create a web application for users to submit their feedback and receive depression severity predictions. The web application collects the input features from users, sends a POST request to the ML API endpoint, and displays the predicted depression severity. We implement the web application using HTML, CSS, and JavaScript, and integrate it with the Flask API.

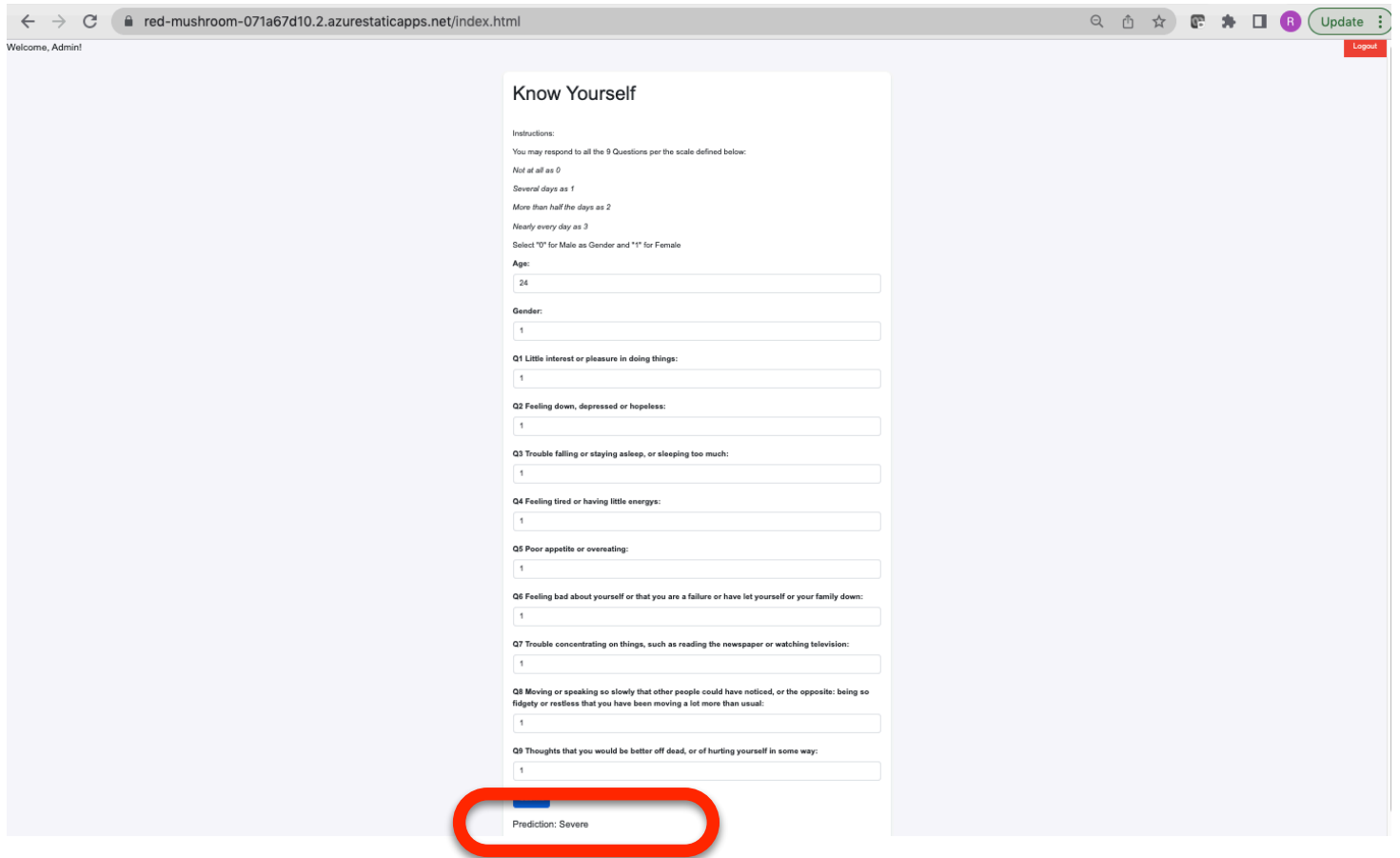
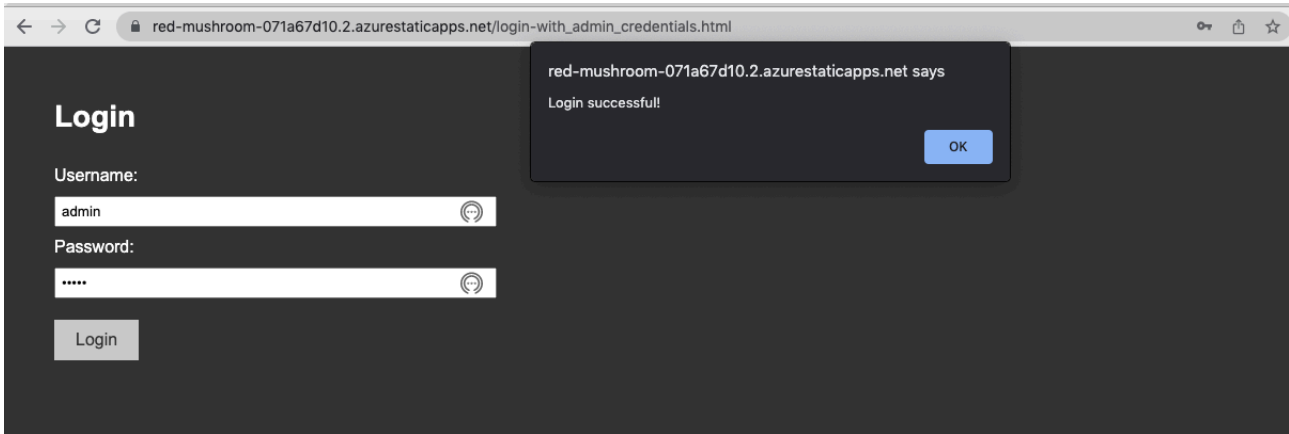
## Deploying the Web Application as a Static App

We deploy the web application as a static app using a service like Azure Static Web Apps or GitHub Pages. This allows us to easily host and scale the web application without managing any servers or infrastructure.

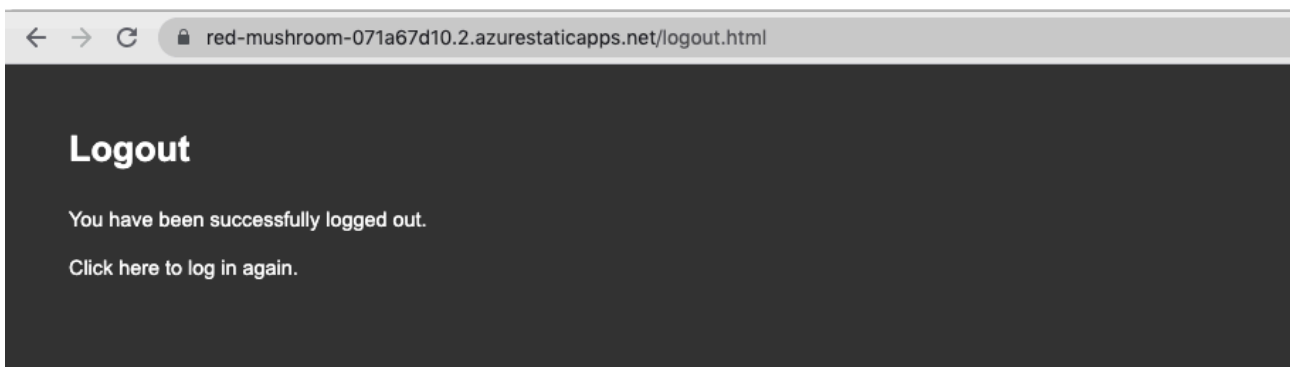
Deployed application can be accessed -

[https://red-mushroom-071a67d10.2.azurestaticapps.net/login-with\\_admin\\_credentials.html](https://red-mushroom-071a67d10.2.azurestaticapps.net/login-with_admin_credentials.html)

Default credentials: admin/admin



As shown above, prediction result printed post click of Submit after filling the assessment form:



## Challenges

1. **Limited and imbalanced data:** Availability of labeled PHQ-9 questionnaire data may be limited, and the distribution of severity levels might be imbalanced, potentially affecting model performance.
2. **Ensuring privacy and security:** Handling sensitive mental health data requires strict adherence to privacy and security protocols.
3. **Model interpretability:** Interpreting machine learning models for clinical use requires transparency and explainability.
4. **Quota issue:** Hampered the deployment of the Auto ML generated model. This led to containerising the model and deploy as API end point using Azure Web App

## Benefits

1. **Improved accuracy:** Azure AutoML's advanced algorithms can yield higher prediction accuracy compared to manual assessments.
2. **Faster assessments:** Automated predictions enable quicker severity assessments, allowing for timely intervention and treatment.
3. **Scalability:** The cloud-based solution can be easily scaled to accommodate larger datasets and serve more users.
4. **Enhanced user experience:** A seamless user interface can increase the adoption and effectiveness of the assessment tool.

## Next steps for this project

This project for predicting depression severity using the PHQ-9 questionnaire and Azure AI can be expanded in several ways in the future:

1. **Incorporate additional data sources:** Integrating data from other mental health assessments, wearables, or social media platforms can potentially improve the model's accuracy and robustness. By considering additional features and data sources, the model can capture more nuances and provide a more comprehensive evaluation of a person's mental health.
2. **Personalized treatment recommendations:** The predictive model can be expanded to suggest personalized treatment plans based on the individual's depression severity, personal preferences, and other relevant factors. This could include recommending therapy types, medication, or lifestyle changes tailored to the individual's needs.
3. **Longitudinal tracking and monitoring:** The system can be enhanced to track individuals' mental health progress over time, allowing for ongoing assessment of depression severity and monitoring of treatment efficacy. This could help in adjusting treatment plans and providing proactive support as needed.
4. **Multilingual support:** Expanding the solution to support multiple languages can make the tool more accessible to a diverse range of users, helping to address mental health issues across different cultural and linguistic contexts.

5. **Integration with mental health care providers:** The system can be integrated with electronic health records (EHRs) or other platforms used by mental health care providers. This would enable seamless sharing of assessment results and facilitate better collaboration between patients and their care teams.

6. **Predicting other mental health disorders:** The approach can be adapted to predict the severity or presence of other mental health disorders, such as anxiety or bipolar disorder, by leveraging relevant questionnaires and datasets.

7. **Expand to mobile and web applications:** Develop mobile and web applications for easy access, enabling users to assess their mental health and track their progress on the go, and allowing mental health professionals to monitor and provide support remotely.

By exploring these expansions, the project can offer a more comprehensive and personalized mental health care experience, ultimately improving the overall effectiveness of mental health interventions and support systems.

## Conclusion

In this blog, we have demonstrated how to build a machine learning model for depression severity prediction using Azure AutoML, Flask, and Docker. We have also discussed the challenges and benefits of our solution, and provided insights into the next steps for improving the model and its deployment. By leveraging these technologies, we can create a powerful and scalable solution that helps improve the detection and treatment of depression for millions of people worldwide.