

Exp No. 7	Creating a Model using AutoML	REG. NO: URK23CS1197
Date:		

Objective:

The main objective of using AutoML (Automated Machine Learning) is to automate the process of developing machine learning models. AutoML tools help in automating tasks like data preprocessing, feature selection, model selection, hyperparameter tuning, and model evaluation, making it easier to create accurate models without requiring extensive machine learning expertise.

Tools/ Software Required:

- mljar-supervised
- fastapi
- uvicorn
- scikit-learn
- joblib

Job Role:

Data Scientist/ML Engineer

Skills Required:

- **Basic Programming Skills:** Familiarity with languages like Python or R.
- **Understanding of ML Concepts:** Knowledge of basic machine learning concepts such as supervised and unsupervised learning, model evaluation metrics, and overfitting/underfitting.
- **Experience with AutoML Tools:** Familiarity with AutoML platforms such as Google AutoML, H2O.ai, Auto-sklearn, or Microsoft Azure AutoML.
- **Data Analysis:** Ability to analyze and preprocess data to ensure it is suitable for modeling.
- **Cloud Services:** Knowledge of cloud platforms (optional but beneficial) for deploying models.

Prerequisites:

- **AutoML Platform Account:** Create an account on a chosen AutoML platform (e.g., Google Cloud AutoML, Azure Machine Learning).
- **Dataset:** A clean and well-prepared dataset to be used for training the model. This includes labeled data for supervised learning tasks.
- **Basic ML Environment Setup:** Install necessary libraries and tools, such as pandas, numpy, scikit-learn, or specific AutoML packages if working locally.

Description:

Automated Machine Learning provides methods and processes to make Machine Learning available for non-Machine Learning experts, to improve efficiency of Machine Learning and to accelerate research on Machine Learning.

AutoML automatically performs the following steps:

- Preprocess and clean the data.
- Select and construct appropriate features.
- Select an appropriate model family.
- Optimize model hyperparameters.
- Design the topology of neural networks (if deep learning is used).
- Postprocess machine learning models.
- Critically analyze the results obtained.

Example: Command-Line Steps (Google Cloud SDK)**Authenticate with Google Cloud:**

```
sh
gcloud auth login
gcloud config set project your-project-id
```

Upload Data to Cloud Storage:

```
sh
gsutil cp local-file-path gs://your-bucket-name/
```

Create and Import Dataset:

```
sh
curl -X POST -H "Content-Type: application/json" \
-H "Authorization: Bearer $(gcloud auth print-access-token)" \
https://automl.googleapis.com/v1/projects/your-project-id/locations/us-central1/datasets \
-d '{
  "displayName": "your-dataset-name",
  "imageClassificationDatasetMetadata": {}
}'
```

Train Model:

```
sh
curl -X POST -H "Content-Type: application/json" \
-H "Authorization: Bearer $(gcloud auth print-access-token)" \
https://automl.googleapis.com/v1/projects/your-project-id/locations/us-central1/models \
-d '{
  "displayName": "your-model-name",
  "datasetId": "your-dataset-id",
  "imageClassificationModelMetadata": {
    "trainBudgetMilliNodeHours": 24000
  }
}'
```

```
}
}'
```

Deploy Model:

```
sh
curl -X POST -H "Content-Type: application/json" \
-H "Authorization: Bearer $(gcloud auth print-access-token)" \

https://automl.googleapis.com/v1/projects/your-project-id/locations/us-central1/models/your-model-id:deploy
```

Questions:

- 1.Design a project that uses AutoML to create and deploy a machine learning model.

Program:

Project Structure:

```
automl-mlops/
├── train_model.py
├── model.pkl
├── model.py
├── app.py
├── AutoML_1/
└── requirements.txt
```

requirements.txt

```
mljar-supervised
fastapi
uvicorn
scikit-learn
joblib
```

train_model.py

```
from supervised.automl import AutoML
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import joblib

data = load_iris()
X = data.data
y = data.target
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
automl = AutoML(mode="Explain", total_time_limit=60)
automl.fit(X_train, y_train)
```

```
y_pred = automl.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print("✅ Model Accuracy:", acc)
```

```
print("\n===== AUTO ML LEADERBOARD (Proof of AutoML) =====")
print(automl.get_leaderboard())
```

```
joblib.dump(automl, "model.pkl")
print("\n✅ Model saved as model.pkl")
```

app.py

```
from fastapi import FastAPI
from pydantic import BaseModel
import joblib
```

```
model = joblib.load("model.pkl")
```

```
app = FastAPI()
```

```
class IrisInput(BaseModel):
    sepal_length: float
    sepal_width: float
    petal_length: float
    petal_width: float
```

```
@app.post("/predict")
def predict(data: IrisInput):
    X = [[data.sepal_length, data.sepal_width, data.petal_length, data.petal_width]]
    result = model.predict(X)[0]
    return {"predicted_class": int(result)}
```

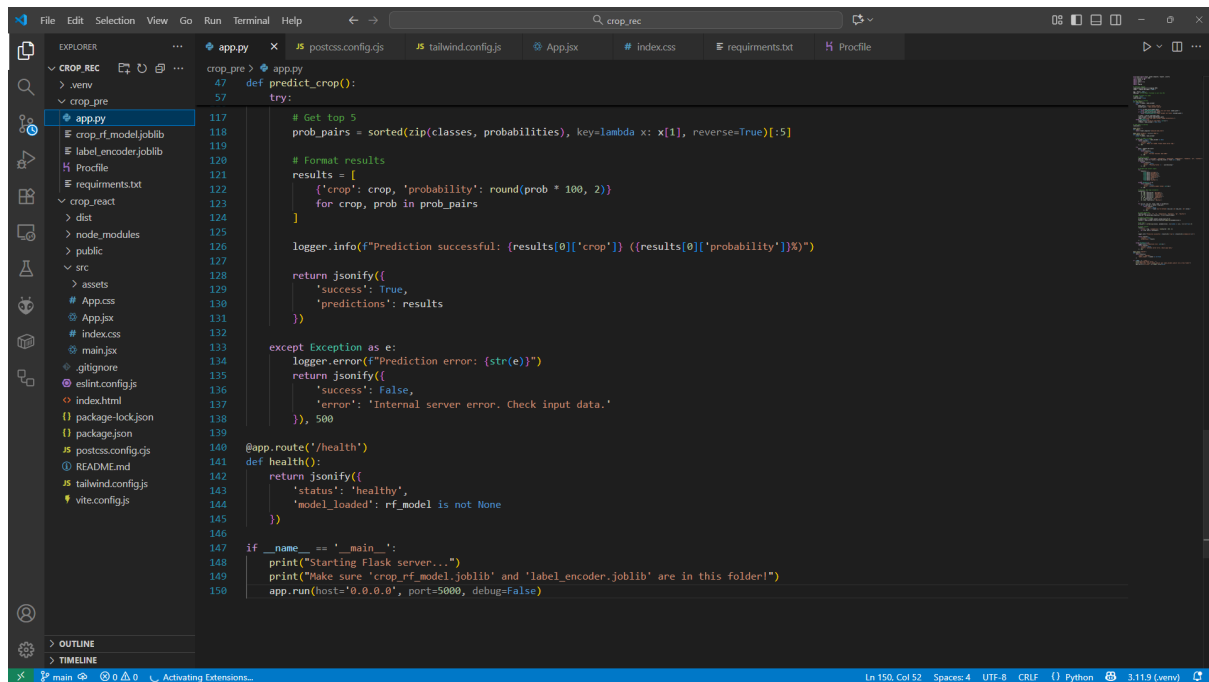
.

Expected Output :

API link: <https://automi-mlops.onrender.com>

Q1.

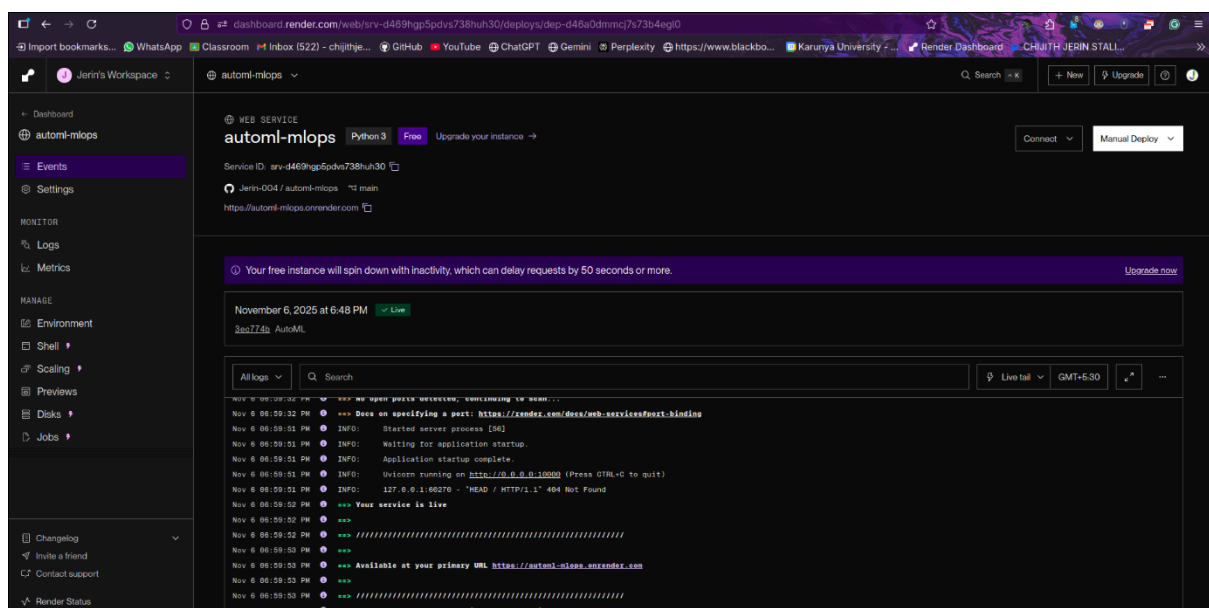
Output of train_model.py (AUTO ML LEADERBOARD)



```

47 def predict_crop():
48     try:
49         # Get top 5
50         prob_pairs = sorted(zip(classes, probabilities), key=lambda x: x[1], reverse=True)[:5]
51
52         # Format results
53         results = [
54             {'crop': crop, 'probability': round(prob * 100, 2)}
55             for crop, prob in prob_pairs
56         ]
57
58         logger.info(f"Prediction successful: {results[0]['crop']} ((results[0]['probability'])*K)")
59
60         return jsonify({
61             'success': True,
62             'predictions': results
63         })
64
65     except Exception as e:
66         logger.error(f"Prediction error: {str(e)}")
67         return jsonify({
68             'success': False,
69             'error': 'Internal server error. Check input data.'
70         }), 500
71
72 @app.route('/health')
73 def health():
74     return jsonify({
75         'status': 'healthy',
76         'model_loaded': rf_model is not None
77     })
78
79 if __name__ == '__main__':
80     print("Starting Flask server...")
81     print("Make sure 'crop_rf_model.joblib' and 'label_encoder.joblib' are in this folder!")
82     app.run(host='0.0.0.0', port=5000, debug=False)
  
```

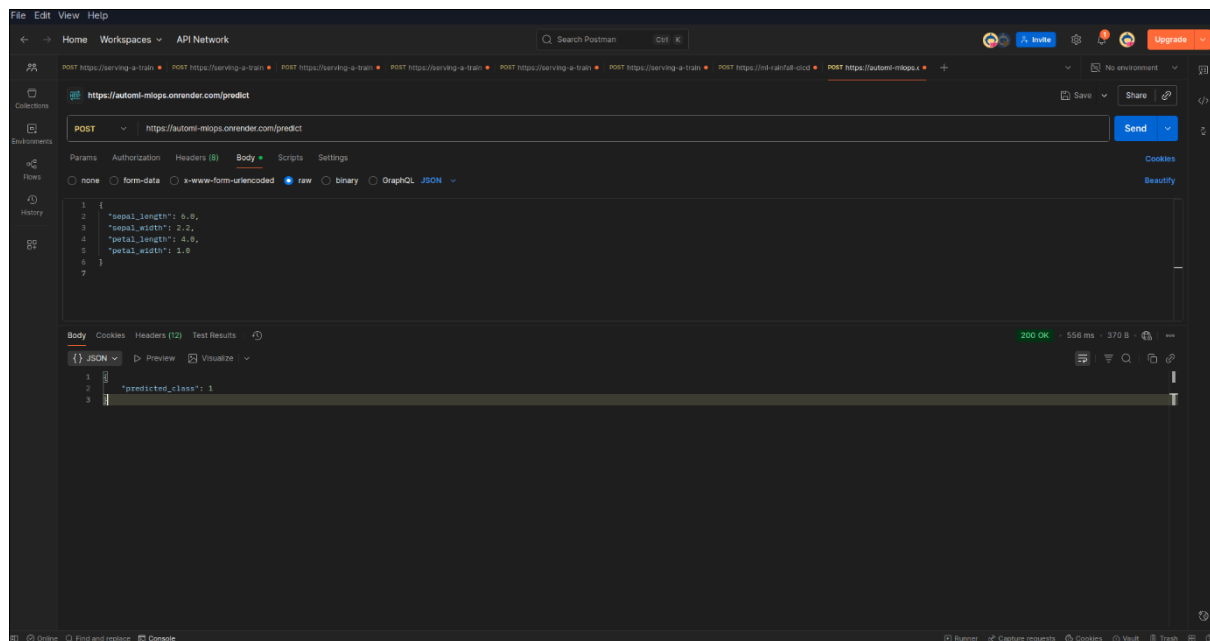
Deployment using Render



```

Nov 6 06:59:32 PM *** Base on specifying a part: https://render.com/docs/web-services#runtime
Nov 6 06:59:31 PM INFO: Started server process [56]
Nov 6 06:59:31 PM INFO: Waiting for application startup.
Nov 6 06:59:31 PM INFO: Application startup complete.
Nov 6 06:59:31 PM INFO: Uvicorn running on http://0.0.0.0:10000 (Press CTRL+C to quit)
Nov 6 06:59:31 PM INFO: 127.0.0.1:60270 - "HEAD / HTTP/1.1" 404 Not Found
Nov 6 06:59:32 PM *** Your service is live
Nov 6 06:59:32 PM ***
Nov 6 06:59:33 PM ***
Nov 6 06:59:33 PM *** Available at your primary URL https://automi-mlops.onrender.com
Nov 6 06:59:33 PM ***
Nov 6 06:59:33 PM ***
Nov 6 06:59:34 PM INFO: 127.0.0.1:60270 - "GET / HTTP/1.1" 404 Not Found
  
```

API testing using Postman



Result :

The AutoML system successfully trained the Iris classification model by automatically selecting the best-performing algorithm and tuning its parameters. The trained model was deployed as a REST API and produced accurate predictions through the /predict endpoint.