

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://automl-mlops.onrender.com>

Q1.

Ouput of train_model.py (AUTO ML LEADERBOARD)

```
File Edit Selection View Go Run Terminal Help ← → search: crop_rec

EXPLORER ... app.py x JS postcss.config.js JS tailwind.config.js App.jsx # index.css requirements.txt profile

CROP_REC ...
  .env
  crop_pre ...
    app.py
    crop_rf.model.joblib
    label_encoder.joblib
    profile
    requirements.txt
  crop_react ...
    dist
    node_modules
    public
    src ...
      assets
      App.css
      App.jsx
      index.css
      main.jsx
      ignore
      eslint.config.js
    index.html
    package-lock.json
    package.json
    postcss.config.js
    README.md
    tailwind.config.js
    vite.config.js

OUTLINE
TIMELINE

Activating Extensions... In 150, Col 52 Spaces: 4 UTF-8 CR LF Python 3.11.9 (venv)
```

```
app.py
117     # Get top 5
118     prob_pairs = sorted(zip(classes, probabilities), key=lambda x: x[1], reverse=True)[:5]
119
120     # Format results
121     results = [
122         {'crop': crop, 'probability': round(prob * 100, 2)}
123         for crop, prob in prob_pairs
124     ]
125
126     logger.info(f"Prediction successful: {results[0]['crop']} ({results[0]['probability']}%)")
127
128     return jsonify({
129         'success': True,
130         'predictions': results
131     })
132
133 except Exception as e:
134     logger.error(f"Prediction error: {str(e)}")
135     return jsonify({
136         'success': False,
137         'error': 'Internal server error. Check input data.'
138     }), 500
139
140 @app.route('/health')
141 def health():
142     return jsonify({
143         'status': 'healthy',
144         'model_loaded': rf_model is not None
145     })
146
147 if __name__ == '__main__':
148     print("Starting Flask server...")
149     print("Make sure 'crop_rf.model.joblib' and 'label_encoder.joblib' are in this folder!")
150     app.run(host='0.0.0.0', port=5000, debug=False)
```

Deployment using Render

The screenshot shows the dashboard.render.com interface for the 'automl-mlops' service. The left sidebar has a purple header 'Jerin's Workspace' and sections for Dashboard, automl-mlops (selected), Events, Settings, MONITOR, Logs, and Metrics. The main area shows the service configuration: 'WEB SERVICE' type, Python 3 runtime, and a 'Free' plan. It includes a 'Connect' button, a 'Manual Deploy' button, and a note about instance spin down. Below is a log viewer with a search bar, showing logs from November 6, 2025, at 6:48 PM. The logs include startup messages like 'INFO: Starting server process [56]' and 'INFO: Waiting for application startup', followed by a 404 Not Found error for a non-existent URL.

API testing using Postman

The screenshot shows the Postman application interface. A POST request is being made to <https://automl-mlops.onrender.com/predict>. The request body contains the following JSON data:

```

1 {
2   "sepal_length": 6.0,
3   "sepal_width": 2.0,
4   "petal_length": 4.0,
5   "petal_width": 1.0
6 }
7
  
```

The response status is 200 OK, with a response time of 556 ms and a response size of 370 B. The response body is a JSON object with one key, "predicted_class", which has a value of 1.

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.