Date

Expt. No. 1 Experiment - 1 Page No. 1

Aim: To Learn about Github and Git, uploading code to Github using Git Commands

Theory :-

- Git : Git is a distributed version control system that allows developers to track changes in code, collaborate with others, and manage different versions of a project.

- GitHub : It is a cloud-based hosting platform for Git repositories. It provides collaboration features like pull requests, issues, and project boards, making teamwork easier.

Basic Git work flow :-

1.) Initialize Git in the project (git init)
2) Add files to staging (git add)
3) Commit changes (git commit)
4) Connect with Github repository
5) Push code to Github (git push)

Teacher's Signature

Expt. No. 1 Date

Page No. 2

Step by step Example :-

Step 1 : Install Git

Download and install git from https://git-scm.com

check installation :

```
git --version
```

Step 2 : Configure Git (only once)

Command :-

```
git config --global user.name "your name"
git config --global user.email "your.email@example.com"
```

Step 3 : Initialize a Git Repository

Navigate to your project folder :

```
cd my-project
git init
```

Step 4 : Add files to staging Area

```
git add.
```

Step 5 : Commit changes

```
git commit -m "First commit-project setup"
```

Teacher's Signature

Date

Expt. No. 1 Page No. 3

Step 6: Create a Repository on Github
1) Log in to Github
2) Click New Repository
3) Give it a name, eg, my-project.
4) Copy the HTTPS URL

Step 7: Link Local Repo to Github
git remote add origin
https://github.com/username/my-project.git

Step 8: Push Code to Github
git branch -M main
git pus -u origin main

Conclusion :-
Git is a version control system to track code changes, while Github is a platform to store and share code online.

Teacher's Signature

Date

Expt. No. 2 Experiment - 2 Page No. 4

Aim : To perform classification using random forest on svm models on a dataset

Data set :

Name - Tele_churn.csv

no. of. colums - 21

no. of. rows - 7043

Theory :-

Random Forest :-

Random forest is a powerful ensemble machine learning algorithm used for classification and regression tasks. It works by combining the predictions of many decision trees to produce a more accurate and stable result.

How it works :-

Random forest builds multiple decision trees during training. Each tree is trained on a random subset of the training data. At each split in a tree, it only considers a random subset of features. For classification, each tree votes for a class and majority

Teacher's Signature

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import StandardScaler

# Load dataset
df = pd.read_csv("data.csv")

# Separate features and target
X = df.drop("target", axis=1)
y = df["target"]

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
# (Optional) Standardize features — Random Forest does NOT require scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize Random Forest
rf = RandomForestClassifier(
    n_estimators=100,      # number of trees
    max_depth=None,       # let trees expand fully
    random_state=42
)
# Train model
rf.fit(X_train, y_train)

# Predict
y_pred = rf.predict(X_test)

# Evaluation
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

wins. for regression, it averages the prediction values of all trees.

Procedure:-

1) Read the dataset/file
2) cKecK if any null values is present
3) checK if datatype conversion is required
4) Delete unwanted columns
5) Applying label encoding
6) Split the data for model
7) Train and validate the data

Conclusion:-

we used random forest classification for this experiment where the model is trained on 80% of training data and tested on remaining 20% of data as default, which resulted in 79.8% of accuracy.

Teacher's Signature

Date

Expt. No. 3 Experiment - 3 Page No. 6

Aim :- To Deploy random forest classification model using mlflow.

Tools & Software :-
Google Colab
ML flow

Dataset :-
Tele_customer_churn.csv

Setup / procedure :-

1) open Google Colab
2) open sketch book
3) connect runtime
4) load the dataset
5) preprocess the dataset
6) Perform experiment

ML flow :-
ML flow is an open-source platform purpose built to assist machine learning practitioners and team in handeling

Teacher's Signature

```python
import mlflow
import mlflow.sklearn
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

df = pd.read_csv("data.csv")

X = df.drop("target", axis=1)
y = df["target"]
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
# MLflow Experiment Tracking
mlflow.set_experiment("RF_Classifier_Experiment")

with mlflow.start_run():
    rf = RandomForestClassifier(
        n_estimators=100,
        max_depth=5,
        random_state=42
    )
    rf.fit(X_train, y_train)

    # Predict + Evaluation
    y_pred = rf.predict(X_test)
    acc = accuracy_score(y_test, y_pred)

    # Log params & metrics
    mlflow.log_param("n_estimators", 100)
    mlflow.log_param("max_depth", 5)
    mlflow.log_metric("accuracy", acc)

    mlflow.sklearn.log_model(rf, "rf_model")

    print("Model logged with accuracy:", acc)
    print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

Date

Expt. No. 3 Page No. 7

the complexities of machine learning process. MLflow focuses on the full lifecycle for machine learning project ensuring that each phase is manageable traceable and reproducible.

Conclusion :-

Random Forest classifier has been deployed successfully with accuracy of 79%. and auc-roc-value os 81. we use MLflow open source and their commands to connect the url with drive to track seamlessly.

Teacher's Signature

Date

Expt. No. 4 Experiment - 4 Page No. 8

Aim :- To implement random forest classifier using automated pipelines.

Pipelines :-

In mlops a pipeline is an automated sequence of steps that takes raw data all the way to build model test, delivery to a deployed, monitored ML model.

Why Pipelines :-

1) Automation :-

Avoids manual error-prone tasks.
eg. re-running, preprocessing scripts.

2) Reproducibility :-

Ensure the code can be used without any change in method.
eg - same data, same transformation, same result.

3) Scalability :-

Handles big data and complex workflow

Teacher's Signature

```python
# model_only_pipeline.py
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Load your dataset (assume already cleaned / preprocessed)
df = pd.read_csv("data.csv")   # must have no missing values, encoded categories, etc.
target = "target"

X = df.drop(columns=[target])
y = df[target]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Pipeline with only the model (no preprocessing)
clf = Pipeline(
    steps=[
        ("model", RandomForestClassifier(n_estimators=200, random_state=42))
    ]
)

# Train & evaluate
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

Expt. No. 4 Date

Page No. 9

consistently.

4) Collaboration :-
prevents mismatch by data scientists, engineers and ops team to work on clear define steps.

5) Consistency :-
Prevents mismatch between training and production environments eg:- different processing.

Conclusion :-
By observing results, we can determine that model effectively predicts non-churn customers but struggles to identify due to class imbalance. Since defecting churn is vital for retention strategies, improvement like handling imbalance, using advance model like xGBoost.

Teacher's Signature

Date

Expt No. 5 Experiment - 5 Page No. 10

Aim : Implement a classification model using weight and Biases.

Theory :-

weight & Biases (W&B) :-

weight & Biases is a machine learning experiment tracking tool that helps log hyperparameters, monitor metrics like loss and accuracy. and visualize training progress in real time. It is widely used in MLops for reproducibility, collaboration and experiment management.

Why W&B :-

- To track experiments automatically
- To enable visualization of training progress in an iteractive dashboard
- To support comparison scores across runs, making hyper parameter tuning easier.

Parts of code & why :-

Initialization :-

wandb.init (project = "mnist_classification")

Teacher's Signature

```python
import pandas as pd
import wandb
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

wandb.init(project="rf-simple")

df = pd.read_csv("data.csv")
target = "target"

X = df.drop(columns=[target])
y = df[target]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

model = RandomForestClassifier(
    n_estimators=100,
    random_state=42
)
wandb.log({"n_estimators": 100})

model.fit(X_train, y_train)

y_pred = model.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print("Accuracy:", acc)

wandb.log({"accuracy": acc})

wandb.sklearn.log_model(model, "rf_simple_model")

wandb.finish()
```

Expt. No. 5 Date

Page No. 11

start a new w&B project

Config storage :-

wandb.config stores hyperparameters.

logging,

wandb.log({"epoch": epoch, "loss": loss})

records loss values each epoch so they appear in the w&B dashboard.

Conclusion :-

Weight & Biases (w&B) was integrated with a simple pytorch model for MNIST classification to demonstrate experiments tracking and monitoring hyperparameters such as learning rate and epoch were stored in wandb.config and training loss was logged at each epoch.

Teacher's Signature

Date

Expt No. 6 Experiment - 6 Page No. 12

Aim : To deploy model using Docker.

Tools used :-

- Python
- Docker
- Flask or Fast API
- Trained ML Model

Theory :-

Docker is a platform that allows developers to package application and their dependencies into containers. Containers ensure that the application runs uniformly across different environments.

Procedures :-

1) Install Docker Desktop and verify, installation using

docker --version

2) Create a project folder containing :

- app.py -> Flask application to serve model predictions.

Teacher's Signature

- model.pkl -> Pre trained ML model file.
- requirements.txt -> List of dependencies (flask, scikit-learn, etc)
- Dockerfile -> Define the container env.

3) Example Dockerfile :-

```
FROM python:3.9
WORKDIR /app
COPY ./app
RUN pip install requirements.txt
EXPOSE 5000
CMD ["python", "app.py"]
```

4) Build the Docker image :-

```
docker build -t ml-model
```

5) Run the Container :-

```
docker run -p 5000:5000 ml-model
```

6) Test the model API by sending request :-

```
curl http://localhost:5000/predict
```

Conclusion :- The ML model successfull containerzed and deployed as a docker container.

Teacher's Signature

Expt. No. 7 Experiment - 7 Page No. 14 Date

Aim: To monitor the ML model using Evidently AI Tool.

Tools Used :-
- Python
- Evidently AI
- Jupyter Notebook / python script

Theory :-
Evidently AI is an open source library used to track data quality, data drift, and model performance over time. It provides interactive dash-board and integrates easily with pipelines.

Procedure :-
1) Install the Evidently library:
pip install evidently

2) Import the library and prepare datasets
from evidently.report import Report
from evidently.metric_preset import DataDriftPreset
report = Report(metrics = [DataDriftPreset()])

Teacher's Signature

Expt. No. 07 Date

Page No. 15

```
report.run(reference_data = ref_df,
           current_data = curr_df)
report.save_html("data_drift_report.html")
```

3) Compare reference and current datasets to detect drift.

4) View the HTML report to analyze data changes and potential model performance issues.

Conclusion :-

The Evidently AI Tool successfully generated a drift and performance report. helping to monitor the model's real-time reliability and detect data inconsistencies.

Teacher's Signature

Expt. No. 8 Experiment - 8 Date Page No. 16

Aim : To visualize system and model metrics using Grafana dashboards.

Tools used :-
- Grafana
- Prometheus
- Web Browser

Theory :-
Grafana is an open source analytics and visualization platform used to monitor time-series data such as CPU usage, latency or ML metrics. It connects to multiple data sources like prometheus, influx DB etc.

Procedure :-
1) Install Grafana :-
sudo apt-get install grafana

2) Start Grafana service :-
sudo systemctl start grafana-server

3) Access Grafana UI at http://localhost:3000
(default credentials : admin/admin)

Teacher's Signature

Expt No. 8 Date

Page No. 17

4) Add Prometheus as a data source.

5) Create dashboard and add panels for metrics such as:
   - Model accuracy
   - Request latency
   - CPU and memory usage

6) Save and share dashboard.

Conclusion :-

Grafana successfully used to visualize and monitor ML model metrics in real-time, improving observability and system health tracking.

Teacher's Signature

Expt No. 9 Experiment - 9 Page No. 18 Date

Aim: To understand and use Kubernetes for deploying scaling and containerized ML applications.

Tools used :-
- Kubernetes (MiniKube or cloud cluster)
- Kubectl CLI
- Docker

Theory :-
Kubernetes (K8s) automates deployment, scaling, and management of containerized applications. It groups containers into logical units called pods, ensuring high availability and fault tolerance.

Procedure :-
1) Install and start MiniKube :
```
minikube start
```
2) Create a Deployment YAML file
```
apiVersion : apps/v1
Kind : Deployment
Metadata :
    name : ML-model-deployment
```

Teacher's Signature

Expt. No. 9 Date

Page No. 19

```
spec:
  replicas: 2
  selector:
    matchLabels:
      app: ML-model
  template:
    metadata:
      labels:
        app: ml-model
    spec:
      containers:
        - name: ml-model
          image: ml-model:latest
          ports:
            - containerPort: 5000
```

3) Deploy and expose the Application

Kubectl apply -f deployment.yaml

Kubectl expose deployment ml-model-deployment --type=NodePort --port=5000

4) Access the model through the cluster endpoint

Conclusion: Kubernetes was successfully used to deploy and manage containerized ML model, enabling scalability and fault-tolerance.

Teacher's Signature

Expt. No. 10 Experiment - 10 Date Page No. 20

Aim : To called and store model performance metrics using Prometheus

Tools used :-
- Prometheus
- Python
- Grafana

Theory :-
Prometheus is a time-series database and monitoring system that scrapes matrics from targets exposed via HTTP endpants It's commonly paired with Grafana for visualization and alerting.

Procedure :-
1) install Prometheus from https://prometheus.io

2) Configure the prometheus.yml file to monitor your flask model.

```
scrap_configs:
  - job_name: 'ml-model'
    static_configs:
      - targets: ['localhost:5000']
```

Teacher's Signature

Expt. No. 10 Date

Page No. 21

3) start Prometheus :-

prometheus --config.file=prometheus.yml

4) Integrate Prometheus client in your ML App :

```
from prometheus_client import
        start_http_server, Summary

REQUEST_TIME =
Summary('request_processing_seconds',
        'time spent processing request')
start_http_server(8000)
```

5) Access prometheus UI at http://localhost:9090

Conclusion :-

Prometheus successfully collected and displayed real-time metrics from the ML model, proving insights into performance and system load.

Teacher's Signature