PAYZONE PREDICTION USING MACHINE LEARNING AND STREAMLIT

Project Report by [Sivashiri M (2022311045]

Objective

The objective of this project is to develop a machine learning-based tool to predict whether a given subsurface geological zone qualifies as a potential oil or gas payzone. The prediction is based on four key reservoir properties: porosity, water saturation, permeability, and net pay thickness.

Data Generation and Preparation

Since real field data was not available for this project, I began by synthetically generating a dataset that mimics realistic reservoir conditions. The dataset included both "payzone" and "non-payzone" cases with labeled outputs based on thresholds commonly used in petroleum engineering, such as net pay thickness being greater than 10 meters.

The following features were chosen for the application:

- Porosity (%)
- Water Saturation (%)
- Permeability (mD)
- Net Pay Thickness (m)

Model Selection and Training

To build the predictive model, I experimented with the Multi-Layer Perceptron (MLP) Classifier

For the MLP model, I applied feature scaling using StandardScaler to normalize input features. Despite showing high cross-validation scores (up to 0.90 F1-score), it underperformed during manual validation, likely due to sensitivity to input distributions.

The model was evaluated using the following metrics:

- F1 Score
- Accuracy
- Confusion Matrix
- Manual input validation

Validation:

The model was then validated separately using the cross-validation method and the manual validation using an external unseen data.

Streamlit App Deployment

I built an interactive Streamlit web application (app.py) where users can input reservoir parameters and instantly get a prediction on whether the zone is a payzone.

The features of the application are as follows:

- Clean, user-friendly UI
- Real-time predictions
- Input validation for numerical values
- Model confidence display (optional)

GitHub Actions Integration

To automate workflow testing, I created a GitHub Actions workflow file (.github/workflows/deploy.yml) that:

- Installs dependencies
- Checks if the Streamlit app runs successfully
- Helps ensure continuous integration readiness

Project Structure

```
payzone-predictor/
├── app.py  # Streamlit web app
├── model.pkl  # Trained ML model
├── scaler.pkl  # Scaler for MLP (optional)
├── requirements.txt  # Python dependencies
└── .github/
    └── workflows/
        └── deploy.yml  # GitHub Actions workflow
```

Learnings and Challenges

- Learned how to handle model scaling and persistence for deployment.
- Understood the importance of manual validation beyond cross-validation metrics.
- Gained hands-on experience in building and deploying Streamlit apps.
- Faced and resolved issues related to module loading, input scaling, and threshold effects.

Conclusion

This project helped me explore how machine learning can be integrated into real-world reservoir evaluation workflows. I now have a working ML-powered payzone predictor hosted as a web app, and a complete GitHub repository with automated testing.