Machine Learning Internship Assignment Report

Submitted by: Sagar Purswani

Date: March 14, 2025

---

1. Introduction

This report outlines the approach taken to complete the given machine learning assignment for ImagoAI's Internship. The objective was to preprocess spectral data, apply dimensionality reduction using PCA (Principal Component Analysis), and predict values using a Random Forest model. The solution was implemented in Python and deployed as an interactive Streamlit application.

---

2. Data Preprocessing

2.1 Loading the Dataset

The dataset provided (TASK-ML-INTERN.csv) was loaded using pandas for analysis. The dataset contained spectral data, including an identifier column (hsi_id).

2.2 Data Cleaning and Feature Selection

- Removed the hsi_id column, as it was not a predictive feature.
- Ensured all features were numerical and checked for missing values.

2.3 Feature Scaling

- Used StandardScaler from sklearn.preprocessing to standardize the dataset.
- Standardization helps improve model performance by ensuring all features are on the same scale.

python

Copy code

```
from sklearn.preprocessing import StandardScaler


scaler = StandardScaler()
X_scaled = scaler.fit_transform(data.iloc[:, 1:-1])  # Excluding ID column
```

---

3. Dimensionality Reduction using PCA

Since the dataset had a high number of features, PCA was applied to reduce dimensionality while preserving variance.

- PCA (Principal Component Analysis) was trained and fitted on the scaled data.
- The trained PCA model was saved as pca_model.pkl for later use.

- Reduced features were then used to train the final model.

python

Copy code

```
from sklearn.decomposition import PCA


pca = PCA(n_components=448)  # Selected based on explained variance
X_pca = pca.fit_transform(X_scaled)
```

---

4. Model Selection and Training

The Random Forest Regressor was chosen after testing multiple models (e.g., TabNet, decision trees). It provided the best performance.

4.1 Model Training

- The Random Forest model was trained on the PCA-transformed data.
- The trained model was saved as random_forest_model.pkl for inference.

python

Copy code

```
from sklearn.ensemble import RandomForestRegressor


model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_pca, y)
```

---

5. Streamlit App Development

A Streamlit-based web app (app.py) was developed to allow users to upload spectral data and get predictions interactively.

5.1 Steps in the App

- Upload a CSV file containing spectral data.
- Preprocess the uploaded data:
    - Remove hsi_id column.
    - Apply StandardScaler for normalization.
    - Transform data using PCA.
- Use the trained Random Forest model for prediction.
- Display the predictions on the interface.

5.2 Handling Edge Cases

- The app checks for missing or extra columns in the uploaded data.
- It ensures that uploaded files are correctly formatted CSVs before making predictions.

python

Copy code

```
import streamlit as st
import pickle
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler

# Load the trained models
@st.cache_resource
def load_models():
    with open("pca_model.pkl", "rb") as f:
        pca = pickle.load(f)
    with open("random_forest_model.pkl", "rb") as f:
        model = pickle.load(f)
    return pca, model

pca, model = load_models()

st.title("Spectral Data Prediction")

uploaded_file = st.file_uploader("Upload a CSV file", type=["csv"])

if uploaded_file:
    data = pd.read_csv(uploaded_file)

    # Remove ID column
    if 'hsi_id' in data.columns:
```

```
    data = data.drop(columns=['hsi_id'])


# Standardization
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)


# PCA Transformation
data_pca = pca.transform(data_scaled)


# Prediction
predictions = model.predict(data_pca)


st.write("Predictions:", predictions)
```

---

6. Results and Evaluation

6.1 Model Performance Metrics

The Random Forest model performed the best among the trained models.

Metric	Value
MAE	4101.92
RMSE	16392.38
R^2 Score	0.0387

These metrics indicate that the model captures some patterns in the data, though further improvements (e.g., hyperparameter tuning, feature engineering) could enhance performance.

---

7. Key Takeaways and Improvements

What Worked Well?

✓ Effective dimensionality reduction using PCA.
✓ Random Forest model outperformed other models.
✓ Interactive Streamlit app for easy predictions.

Areas for Improvement

- 🔹 Further hyperparameter tuning could improve accuracy.
- 🔹 Additional feature engineering might help extract more insights.
- 🔹 Using pre-trained spectral models could enhance performance.

---

8. Repository Overview

GitHub Repository: [Your Repo Link]

Files Included:

📌 Codes_SpectralPrediction.ipynb → Notebook with data analysis, training, and evaluation.
📌 app.py → Streamlit app for interactive predictions.
📌 pca_model.pkl → PCA model used for dimensionality reduction.
📌 random_forest_model.pkl → Trained Random Forest regression model.
📌 TASK-ML-INTERN.csv → Provided dataset.
📌 README.md → Installation steps and project overview.
📌 Report.pdf → This document summarizing the project.

---

9. Conclusion

This project successfully implemented spectral data processing, dimensionality reduction, and regression modeling for prediction tasks. The Streamlit app allows real-time user interaction, making it an easy-to-use tool for spectral analysis.

Further improvements can be made by tuning hyperparameters, exploring deep learning models, or incorporating domain-specific knowledge into feature engineering.

Thank you for reviewing my work! Looking forward to your feedback.