l3p0fkite

February 7, 2025

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import accuracy_score, classification_report,␣
  ↪confusion_matrix
```

[3]:
```
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.naive_bayes import GaussianNB
```

[4]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
```

1 Loading the Dataset

[5]:
```
dataset_path = "/content/sr_wq_rs_join.csv"
df = pd.read_csv(dataset_path, encoding='latin1')
```

[6]:
```
unnecessary_columns = ['system:index', 'SiteID', 'date_unity', 'path', 'row',␣
  ↪'sat', '.geo', 'endtime', 'date', 'date_only', 'source', 'lat', 'long',␣
  ↪'TZID', 'date_utc', 'clouds', 'time', 'landsat_id', 'timediff', 'pwater',␣
  ↪'type', 'id']
df = df.drop(columns=unnecessary_columns, errors='ignore')
```

[7]:
```
df.fillna(df.median(), inplace=True)
```

2 Feature Selection

[8]:
```
features = ['chl_a', 'doc', 'secchi', 'tss', 'p_sand']
```

[9]:

```
def classify_suitability(value):
    if value >= 0.7:
        return "Highly Preferred"
    elif 0.4 <= value < 0.7:
        return "Partially Preferred"
    else:
        return "Least Preferred"
```

[10]:

```
df['water_suitability'] = (df['chl_a'] * 0.3 + df['secchi'] * 0.3 + df['doc'] *␣
 ↪0.2 + df['tss'] * 0.2) / 4  # Example formula
df['agriculture_suitability'] = df['water_suitability'].
 ↪apply(classify_suitability)
```

3 Label Encoding

[11]:

```
label_encoder = LabelEncoder()
df['suitability_encoded'] = label_encoder.
 ↪fit_transform(df['agriculture_suitability'])
```

[12]:

```
X = df[features]
y = df['suitability_encoded']
```

4 Model Training

[13]:

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

[14]:

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

5 Random Forest

[15]:

```
rf_model = RandomForestClassifier(n_estimators=200, max_depth=10,␣
 ↪random_state=42)
rf_model.fit(X_train, y_train)
rf_preds = rf_model.predict(X_test)
print("Random Forest Accuracy:", accuracy_score(y_test, rf_preds))
```

```
Random Forest Accuracy: 0.999187982135607
```

6 XGBoost Model

[16]:
```
xgb_model = XGBClassifier(n_estimators=300, learning_rate=0.05, max_depth=10)
xgb_model.fit(X_train, y_train)
xgb_preds = xgb_model.predict(X_test)
print("XGBoost Accuracy:", accuracy_score(y_test, xgb_preds))
```

```
XGBoost Accuracy: 0.99944484492944 56
```

7 Naive Bayes Model

[17]:
```
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
nb_preds = nb_model.predict(X_test)
print("Naive Bayes Accuracy:", accuracy_score(y_test, nb_preds))
```

```
Naive Bayes Accuracy: 0.9836270683669326
```

8 Neural Networks

[19]:
```
nn_model = Sequential()
nn_model.add(Dense(128, activation='relu', input_shape=(X_train.shape[1],)))
nn_model.add(Dropout(0.3))
nn_model.add(Dense(64, activation='relu'))
nn_model.add(Dropout(0.3))
nn_model.add(Dense(32, activation='relu'))
nn_model.add(Dense(3, activation='softmax'))
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

[21]:
```
from tensorflow.keras.callbacks import EarlyStopping
```

9 Optimization

[22]:
```
nn_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',␣
 ↪metrics=['accuracy'])
early_stopping = EarlyStopping(monitor='val_loss', patience=5,␣
 ↪restore_best_weights=True)
nn_model.fit(X_train, y_train, epochs=30, batch_size=32,␣
 ↪validation_data=(X_test, y_test), callbacks=[early_stopping])
```

```
Epoch 1/30
15086/15086                56s 3ms/step
- accuracy: 0.9980 - loss: 0.0053 - val_accuracy: 0.9989 - val_loss: 0.0029
Epoch 2/30
15086/15086                77s 3ms/step
- accuracy: 0.9982 - loss: 0.0056 - val_accuracy: 0.9982 - val_loss: 0.0042
Epoch 3/30
15086/15086                80s 3ms/step
- accuracy: 0.9981 - loss: 0.0050 - val_accuracy: 0.9988 - val_loss: 0.0033
Epoch 4/30
15086/15086                84s 3ms/step
- accuracy: 0.9981 - loss: 0.0049 - val_accuracy: 0.9987 - val_loss: 0.0032
Epoch 5/30
15086/15086                51s 3ms/step
- accuracy: 0.9983 - loss: 0.0045 - val_accuracy: 0.9990 - val_loss: 0.0027
Epoch 6/30
15086/15086                82s 3ms/step
- accuracy: 0.9984 - loss: 0.0052 - val_accuracy: 0.9991 - val_loss: 0.0026
Epoch 7/30
15086/15086                51s 3ms/step
- accuracy: 0.9981 - loss: 0.0048 - val_accuracy: 0.9990 - val_loss: 0.0035
Epoch 8/30
15086/15086                47s 3ms/step
- accuracy: 0.9984 - loss: 0.0050 - val_accuracy: 0.9991 - val_loss: 0.0028
Epoch 9/30
15086/15086                52s 3ms/step
- accuracy: 0.9984 - loss: 0.0042 - val_accuracy: 0.9987 - val_loss: 0.0030
Epoch 10/30
15086/15086                52s 3ms/step
- accuracy: 0.9983 - loss: 0.0046 - val_accuracy: 0.9985 - val_loss: 0.0037
Epoch 11/30
15086/15086                81s 3ms/step
- accuracy: 0.9984 - loss: 0.0047 - val_accuracy: 0.9982 - val_loss: 0.0041
```

```
[22]: <keras.src.callbacks.history.History at 0x7d97994c5190>
```

10 Evaluation of the Model

```
[23]: nn_preds = np.argmax(nn_model.predict(X_test), axis=1)
      print("Neural Network Accuracy:", accuracy_score(y_test, nn_preds))
```

```
3772/3772              7s 2ms/step
Neural Network Accuracy: 0.9991382667561544
```

11 Classification Report

```
[24]: print("\nRandom Forest Classification Report:\n", classification_report(y_test,␣
       ↪rf_preds))
      print("\nXGBoost Classification Report:\n", classification_report(y_test,␣
       ↪xgb_preds))
      print("\nNaive Bayes Classification Report:\n", classification_report(y_test,␣
       ↪nb_preds))
      print("\nNeural Network Classification Report:\n",␣
       ↪classification_report(y_test, nn_preds))
```

```
Random Forest Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00    119601
           1       0.88      0.35      0.50        20
           2       0.96      0.94      0.95      1066

    accuracy                           1.00    120687
   macro avg       0.95      0.76      0.82    120687
weighted avg       1.00      1.00      1.00    120687


XGBoost Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00    119601
           1       0.92      0.60      0.73        20
           2       0.97      0.97      0.97      1066

    accuracy                           1.00    120687
   macro avg       0.96      0.86      0.90    120687
weighted avg       1.00      1.00      1.00    120687


Naive Bayes Classification Report:
               precision    recall  f1-score   support

           0       1.00      0.99      0.99    119601
           1       0.04      1.00      0.08        20
           2       0.29      0.59      0.39      1066

    accuracy                           0.98    120687
   macro avg       0.45      0.86      0.49    120687
weighted avg       0.99      0.98      0.99    120687
```

```
Neural Network Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    119601
           1       0.00      0.00      0.00        20
           2       0.96      0.94      0.95      1066

    accuracy                           1.00    120687
   macro avg       0.65      0.65      0.65    120687
weighted avg       1.00      1.00      1.00    120687

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```