FA24-CS634101 Data Mining

Final Term Project Report

Professor: Yasser Abduallah

Student: Songjiang Liu

UCID: sl947

GitHub Repository: https://github.com/youjustlook/CS634_songjiang_liu_finaltermproj

Introduction

This report uses three algorithms, namely random forest, decision tree, and convolutional neural network, to do binary prediction and then use several performance evaluation metrics to compare their performance and propose possible explanations for their results. The data set used is from https://archive.ics.uci.edu/dataset/222/bank+marketing, which contains marketing campaign data from a Portuguese bank, and the target is to predict whether the client would subscribe to a term deposit.

About Algorithms Used

Random Forest: an ensemble algorithm which combines multiple decision trees to improve accuracy, robustness, and generalization.
Decision Tree: a tree-like model to split data into branches based on feature conditions to make predictions or classify data.
Convolutional Neural Network: a deep learning algorithm uses filters (convolutions) to detect patterns, which are then combined to identify more complex structures in the data.

Step 0: How to Run this program / Install Necessary Packages

The below packages are required to run the code for using random forest, decision tree, and converlutional neural network algorithms. The metrics used to evaluate the performance of the aforementioned algorithms are implemented manually with self-built functions. Run the following command to install the required packages if missing. Be sure to remove "#" before running the command

In [1]:
```
# Run the following command to install the required packages if missing. Be sure to remove "#" before running the com

# pip install pandas numpy scikit-learn tensorflow seaborn matplotlib
```

Step 1. Data Preparation, Metrics Calculation, and Tabular Display

In [2]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import KFold
from sklearn.metrics import brier_score_loss

import warnings
warnings.filterwarnings("ignore")
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

print("\n-------------------\n\nData Preparation Started...")

data = pd.read_csv("bank.csv", delimiter=";")

# Encode categorical features
categorical_columns = ["job", "marital", "education", "default", "housing", "loan", "contact", "month", "poutcome"]
for col in categorical_columns:
    data[col] = LabelEncoder().fit_transform(data[col])

# Encode target
data['y'] = LabelEncoder().fit_transform(data['y'])

# Normalize
numerical_columns = ["age", "balance", "day", "duration", "campaign", "pdays", "previous"]
scaler = StandardScaler()
data[numerical_columns] = scaler.fit_transform(data[numerical_columns])
```

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

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=66)

# Setting up KFold
n_splits = 10
kf = KFold(n_splits=n_splits, shuffle=True, random_state=66)

print("\n------------------\n\nData Preparation Finished...")

def calculate_metrics(cm, y_true=None, y_prob=None):
    TN, FP, FN, TP = cm.ravel()
    P = TP + FN
    N = TN + FP
    TPR = TP / P
    TNR = TN / N
    FPR = FP / N
    FNR = FN / P
    Recall = TPR
    Precision = TP / (TP + FP)
    F1 = 2 * (Precision * Recall) / (Precision + Recall)
    Accuracy = (TP + TN) / (P + N)
    Error_Rate = (FP + FN) / (P + N)
    BACC = (TPR + TNR) / 2
    TSS = TPR - FPR
    HSS = 2 * (TP * TN - FP * FN) / ((TP + FN) * (FN + TN) + (TP + FP) * (FP + TN))

    # Calculate Brier Score and Brier Skill Score if probabilities are provided
    if y_true is not None and y_prob is not None:
        BS = brier_score_loss(y_true, y_prob)
        BS_ref = brier_score_loss(y_true, [y_true.mean()] * len(y_true))
        BSS = 1 - (BS / BS_ref)
    else:
        BS = None
        BSS = None

    return {
        'TP': TP, 'TN': TN, 'FP': FP, 'FN': FN, 'P': P, 'N': N,
```

```
        'TPR': TPR, 'TNR': TNR, 'FPR': FPR, 'FNR': FNR,
        'Recall': Recall, 'Precision': Precision, 'F1': F1,
        'Accuracy': Accuracy, 'Error Rate': Error_Rate,
        'BACC': BACC, 'TSS': TSS, 'HSS': HSS,
        'Brier Score': BS, 'Brier Skill Score': BSS
    }

# Function to display metrics
def display_metrics(metrics_list, model_name):
    metrics_df = pd.DataFrame(metrics_list).set_index(' ').T
    metrics_df['Average'] = metrics_df.mean(axis=1)
    metrics_df = metrics_df.round(2)
    print(f"\n-------------------\n{model_name} - Metrics for Each Fold and Average of 10 Folds")
    print(metrics_df.to_string())
    return metrics_df
```

```
-------------------

Data Preparation Started...

-------------------

Data Preparation Finished...
```

Step 2. 10-Fold Validation on 3 Algorithms: Random Forest, Decision Tree, and ConvNet1D

In [3]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier
import pandas as pd
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input, Conv1D, GlobalMaxPooling1D, Dense, Dropout
from sklearn.metrics import confusion_matrix

dt_metrics_list = []
rf_metrics_list = []
conv1d_metrics_list = []

print("\n-------------------\n\nModel Running...")
# Perform K-Fold Cross Validation
```

```python
for i, (train_index, test_index) in enumerate(kf.split(X), start=1):
    # Splitting data
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
    X_train_cnn = np.expand_dims(X_train, axis=2)
    X_test_cnn = np.expand_dims(X_test, axis=2)

    # Model Training - Random Forest
    rf_model = RandomForestClassifier(n_estimators=100, random_state=66)
    rf_model.fit(X_train, y_train)

    rf_preds = rf_model.predict(X_test)
    rf_probs = rf_model.predict_proba(X_test)[:, 1]  # Get probabilities for the positive class

    # Calculate and store metrics
    rf_cm = confusion_matrix(y_test, rf_preds)
    rf_metrics = calculate_metrics(rf_cm, y_true=y_test, y_prob=rf_probs)
    rf_metrics[' '] = f'fold_{i}'
    rf_metrics_list.append(rf_metrics)

    # Model Training - Decision Tree
    dt_model = DecisionTreeClassifier(random_state=66)
    dt_model.fit(X_train, y_train)
    # Predictions and Probabilities
    dt_preds = dt_model.predict(X_test)
    dt_probs = dt_model.predict_proba(X_test)[:, 1]  # Get probabilities for the positive class

    # Calculate and store metrics
    dt_cm = confusion_matrix(y_test, dt_preds)
    dt_metrics = calculate_metrics(dt_cm, y_true=y_test, y_prob=dt_probs)
    dt_metrics[' '] = f'fold_{i}'  # Label each fold
    dt_metrics_list.append(dt_metrics)

    # Model Training - Conv1D
    # Conv1D Model Definition
    conv1d_model = Sequential()
    conv1d_model.add(Input(shape=(X_train_cnn.shape[1], 1)))  # Define input shape using Input layer
    conv1d_model.add(Conv1D(64, kernel_size=4, activation='relu'))
    conv1d_model.add(GlobalMaxPooling1D())
    conv1d_model.add(Dropout(0.3))
    conv1d_model.add(Dense(64, activation='relu'))
    conv1d_model.add(Dropout(0.3))
```

```
    conv1d_model.add(Dense(1, activation='sigmoid'))  # Sigmoid for binary classification

    # Compile and Train Model
    conv1d_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    conv1d_model.fit(X_train_cnn, y_train, epochs=5, batch_size=64, verbose=0);

    # Predictions and Confusion Matrix
    conv1d_preds = (conv1d_model.predict(X_test_cnn) > 0.5).astype(int).flatten();
    conv1d_probs = conv1d_model.predict(X_test_cnn).flatten()
    conv1d_cm = confusion_matrix(y_test, conv1d_preds)

    # Calculate and store metrics
    conv1d_metrics = calculate_metrics(conv1d_cm, y_true=y_test, y_prob=conv1d_probs)
    conv1d_metrics[' '] = f'fold_{i}'  # Label each fold
    conv1d_metrics_list.append(conv1d_metrics)

print("\n-------------------\n\nModel Running Finished...")

# Display metrics and capture returned DataFrames with 'Average' column
rf_metrics_df = display_metrics(rf_metrics_list, "Random Forest")
dt_metrics_df = display_metrics(dt_metrics_list, "Decision Tree")
conv1d_metrics_df = display_metrics(conv1d_metrics_list, "Conv1D")

# Combine and print the Average columns in one line
print("\n-------------------\n\nCombined Average Results of Random Forest, Decision Tree, and Conv1D\n",
      pd.DataFrame({
          'Random Forest': rf_metrics_df['Average'],
          'Decision Tree': dt_metrics_df['Average'],
          'Conv1D': conv1d_metrics_df['Average']
      }).round(2).to_string())
```

```
--------------------

Model Running...
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 912us/step
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step

--------------------

Model Running Finished...

--------------------
Random Forest - Metrics for Each Fold and Average of 10 Folds
```

	fold_1	fold_2	fold_3	fold_4	fold_5	fold_6	fold_7	fold_8	fold_9	fold_10	Average
TP	17.00	15.00	14.00	15.00	17.00	20.00	14.00	15.00	18.00	16.00	16.10
TN	401.00	387.00	391.00	398.00	400.00	384.00	388.00	395.00	374.00	387.00	390.50
FP	1.00	10.00	7.00	10.00	4.00	14.00	11.00	13.00	18.00	7.00	9.50
FN	34.00	40.00	40.00	29.00	31.00	34.00	39.00	29.00	42.00	42.00	36.00
P	51.00	55.00	54.00	44.00	48.00	54.00	53.00	44.00	60.00	58.00	52.10
N	402.00	397.00	398.00	408.00	404.00	398.00	399.00	408.00	392.00	394.00	400.00
TPR	0.33	0.27	0.26	0.34	0.35	0.37	0.26	0.34	0.30	0.28	0.31
TNR	1.00	0.97	0.98	0.98	0.99	0.96	0.97	0.97	0.95	0.98	0.98
FPR	0.00	0.03	0.02	0.02	0.01	0.04	0.03	0.03	0.05	0.02	0.02
FNR	0.67	0.73	0.74	0.66	0.65	0.63	0.74	0.66	0.70	0.72	0.69
Recall	0.33	0.27	0.26	0.34	0.35	0.37	0.26	0.34	0.30	0.28	0.31

Precision	0.94	0.60	0.67	0.60	0.81	0.59	0.56	0.54	0.50	0.70	0.65
F1	0.49	0.37	0.37	0.43	0.49	0.45	0.36	0.42	0.37	0.40	0.42
Accuracy	0.92	0.89	0.90	0.91	0.92	0.89	0.89	0.91	0.87	0.89	0.90
Error Rate	0.08	0.11	0.10	0.09	0.08	0.11	0.11	0.09	0.13	0.11	0.10
BACC	0.67	0.62	0.62	0.66	0.67	0.67	0.62	0.65	0.63	0.63	0.64
TSS	0.33	0.25	0.24	0.32	0.34	0.34	0.24	0.31	0.25	0.26	0.29
HSS	0.46	0.32	0.33	0.39	0.46	0.40	0.31	0.37	0.31	0.35	0.37
Brier Score	0.06	0.07	0.07	0.06	0.06	0.07	0.08	0.06	0.08	0.08	0.07
Brier Skill Score	0.38	0.34	0.29	0.33	0.35	0.33	0.23	0.28	0.29	0.30	0.31

Decision Tree - Metrics for Each Fold and Average of 10 Folds

	fold_1	fold_2	fold_3	fold_4	fold_5	fold_6	fold_7	fold_8	fold_9	fold_10	Average
TP	18.00	21.00	22.00	18.00	22.00	19.00	19.00	22.00	31.00	24.00	21.60
TN	375.00	372.00	368.00	375.00	381.00	366.00	369.00	379.00	346.00	359.00	369.00
FP	27.00	25.00	30.00	33.00	23.00	32.00	30.00	29.00	46.00	35.00	31.00
FN	33.00	34.00	32.00	26.00	26.00	35.00	34.00	22.00	29.00	34.00	30.50
P	51.00	55.00	54.00	44.00	48.00	54.00	53.00	44.00	60.00	58.00	52.10
N	402.00	397.00	398.00	408.00	404.00	398.00	399.00	408.00	392.00	394.00	400.00
TPR	0.35	0.38	0.41	0.41	0.46	0.35	0.36	0.50	0.52	0.41	0.42
TNR	0.93	0.94	0.92	0.92	0.94	0.92	0.92	0.93	0.88	0.91	0.92
FPR	0.07	0.06	0.08	0.08	0.06	0.08	0.08	0.07	0.12	0.09	0.08
FNR	0.65	0.62	0.59	0.59	0.54	0.65	0.64	0.50	0.48	0.59	0.58
Recall	0.35	0.38	0.41	0.41	0.46	0.35	0.36	0.50	0.52	0.41	0.42
Precision	0.40	0.46	0.42	0.35	0.49	0.37	0.39	0.43	0.40	0.41	0.41
F1	0.38	0.42	0.42	0.38	0.47	0.36	0.37	0.46	0.45	0.41	0.41
Accuracy	0.87	0.87	0.86	0.87	0.89	0.85	0.86	0.89	0.83	0.85	0.86
Error Rate	0.13	0.13	0.14	0.13	0.11	0.15	0.14	0.11	0.17	0.15	0.14
BACC	0.64	0.66	0.67	0.66	0.70	0.64	0.64	0.71	0.70	0.66	0.67
TSS	0.29	0.32	0.33	0.33	0.40	0.27	0.28	0.43	0.40	0.32	0.34
HSS	0.30	0.34	0.34	0.31	0.41	0.28	0.29	0.40	0.36	0.32	0.34
Brier Score	0.13	0.13	0.14	0.13	0.11	0.15	0.14	0.11	0.17	0.15	0.14
Brier Skill Score	-0.33	-0.22	-0.30	-0.49	-0.14	-0.41	-0.37	-0.28	-0.44	-0.36	-0.33

Conv1D - Metrics for Each Fold and Average of 10 Folds

	fold_1	fold_2	fold_3	fold_4	fold_5	fold_6	fold_7	fold_8	fold_9	fold_10	Average
TP	0.00	1.00	0.00	1.00	0.00	3.00	0.00	0.00	0.00	1.00	0.60
TN	402.00	396.00	398.00	407.00	404.00	396.00	399.00	408.00	392.00	394.00	399.60
FP	0.00	1.00	0.00	1.00	0.00	2.00	0.00	0.00	0.00	0.00	0.40
FN	51.00	54.00	54.00	43.00	48.00	51.00	53.00	44.00	60.00	57.00	51.50
P	51.00	55.00	54.00	44.00	48.00	54.00	53.00	44.00	60.00	58.00	52.10

N	402.00	397.00	398.00	408.00	404.00	398.00	399.00	408.00	392.00	394.00	400.00
TPR	0.00	0.02	0.00	0.02	0.00	0.06	0.00	0.00	0.00	0.02	0.01
TNR	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00	1.00
FPR	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00
FNR	1.00	0.98	1.00	0.98	1.00	0.94	1.00	1.00	1.00	0.98	0.99
Recall	0.00	0.02	0.00	0.02	0.00	0.06	0.00	0.00	0.00	0.02	0.01
Precision	NaN	0.50	NaN	0.50	NaN	0.60	NaN	NaN	NaN	1.00	0.65
F1	NaN	0.04	NaN	0.04	NaN	0.10	NaN	NaN	NaN	0.03	0.05
Accuracy	0.89	0.88	0.88	0.90	0.89	0.88	0.88	0.90	0.87	0.87	0.89
Error Rate	0.11	0.12	0.12	0.10	0.11	0.12	0.12	0.10	0.13	0.13	0.11
BACC	0.50	0.51	0.50	0.51	0.50	0.53	0.50	0.50	0.50	0.51	0.51
TSS	0.00	0.02	0.00	0.02	0.00	0.05	0.00	0.00	0.00	0.02	0.01
HSS	0.00	0.03	0.00	0.04	0.00	0.08	0.00	0.00	0.00	0.03	0.02
Brier Score	0.09	0.09	0.09	0.08	0.08	0.09	0.09	0.08	0.10	0.10	0.09
Brier Skill Score	0.12	0.13	0.16	0.09	0.11	0.12	0.10	0.12	0.12	0.13	0.12

Combined Average Results of Random Forest, Decision Tree, and Conv1D

	Random Forest	Decision Tree	Conv1D
TP	16.10	21.60	0.60
TN	390.50	369.00	399.60
FP	9.50	31.00	0.40
FN	36.00	30.50	51.50
P	52.10	52.10	52.10
N	400.00	400.00	400.00
TPR	0.31	0.42	0.01
TNR	0.98	0.92	1.00
FPR	0.02	0.08	0.00
FNR	0.69	0.58	0.99
Recall	0.31	0.42	0.01
Precision	0.65	0.41	0.65
F1	0.42	0.41	0.05
Accuracy	0.90	0.86	0.89
Error Rate	0.10	0.14	0.11
BACC	0.64	0.67	0.51
TSS	0.29	0.34	0.01
HSS	0.37	0.34	0.02
Brier Score	0.07	0.14	0.09
Brier Skill Score	0.31	-0.33	0.12

Discussion of Results

Random Forest: Achieved the best overall balance between sensitivity (TPR = 0.31), specificity (TNR = 0.98), and accuracy (90%). It also had the highest precision (0.65) and F1 score (0.42). Error rate and Brier score were relatively low, indicating robust performance.

Decision Tree: Demonstrated moderate performance with a slightly better sensitivity (TPR = 0.42) compared to Random Forest but lower specificity (TNR = 0.92). Its accuracy (86%) and precision (0.41) were inferior to Random Forest, and the higher Brier score (0.14) reflects less reliability.

Convolutional Neural Network: Performed poorly for true positive detection (TP = 0.10, TPR = 0.00) but excelled in specificity (TNR = 1.00). Its accuracy (88%) was decent due to the high true negative detection but failed to balance positive predictions, leading to the lowest F1 score (0.04).

Which Performed Better and Why

Random Forest performed better overall due to its superior balance between sensitivity, specificity, and accuracy. Its ensemble approach mitigates overfitting, making it more robust in various scenarios. It aggregates predictions from multiple decision trees, effectively reducing variance and improving generalization. The higher Brier Skill Score (0.31) further emphasizes its reliability and predictive skill in scenarios with class imbalance or rare events. Decision Tree is relatively poorly performing as it suffers from overfitting to the training data and its inability to handle complexity. Convolutional Neural Network failed to detect true positives effectively, likely due to insufficient data for the deep learning model or improper tuning, which is critical for neural networks.