Лабораторна робота №5

ДОСЛІДЖЕННЯ МЕТОДІВ АНСАМБЛЕВОГО НАВЧАННЯ

Мета: використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити методи ансамблів у машинному навчанні.

Хід роботи

Завдання 2.2. Обробка дисбалансу класів

Використовуючи для аналізу дані, які містяться у файлі data_imbalance.txt проведіть обробку з урахуванням дисбалансу класів.

Лістинг коду:

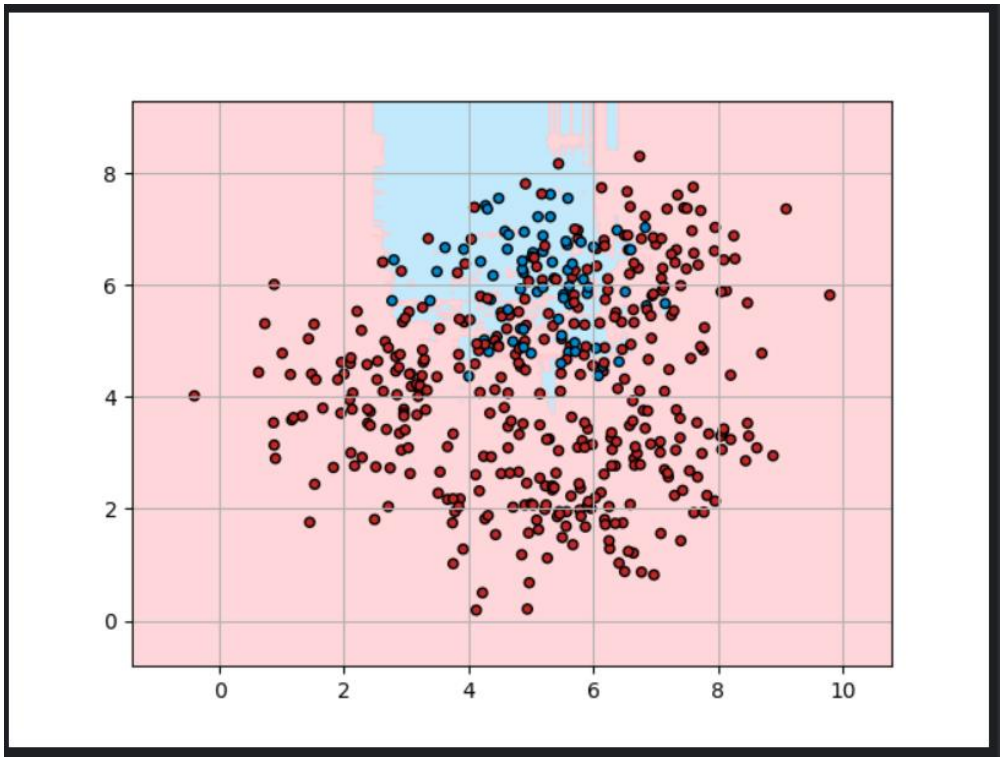
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
from imblearn.over_sampling import SMOTE
from collections import Counter

# 1. Завантаження даних
data = np.loadtxt('data_imbalance.txt', delimiter=",")
X, y = data[:, :-1], data[:, -1]

# Перевірка дисбалансу класів
print("Розподіл класів до обробки:", Counter(y))

# 2. Розділення на тренувальні та тестові дані
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# 3. Використання SMOTE для балансування класів
smote = SMOTE(random_state=42)
X_train_balanced, y_train_balanced = smote.fit_resample(X_train, y_train)

# Перевірка нового розподілу класів
print("Розподіл класів після SMOTE:", Counter(y_train_balanced))

# 4. Створення і навчання класифікатора
clf = RandomForestClassifier(n_estimators=100, random_state=42,
class_weight='balanced')
clf.fit(X_train_balanced, y_train_balanced)

# 5. Оцінка результатів
y_pred = clf.predict(X_test)
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```python
# 6. Візуалізація результатів (якщо дані двовимірні)
def visualize_classifier(classifier, X, y):
    """
    Візуалізація меж класифікації для двовимірних даних.
    """
    from matplotlib.colors import ListedColormap

    # Задаємо нові кольори
    background_colors = ['#B3E5FC', '#FFCDD2']  # Світло-блакитний та світло-червоний
    point_colors = ['#0288D1', '#C62828']  # Темно-синій та темно-червоний

    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01),
                         np.arange(y_min, y_max, 0.01))

    Z = classifier.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    # Фон для меж класифікації
    plt.contourf(xx, yy, Z, alpha=0.8, cmap=ListedColormap(background_colors))

    # Точки даних
    plt.scatter(X[:, 0], X[:, 1], c=y, edgecolor='k', s=20,
                cmap=ListedColormap(point_colors))

    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.grid(True)
    plt.show()

# Візуалізація, якщо у даних є лише 2 ознаки
if X_train.shape[1] == 2:
    visualize_classifier(clf, X_test, y_test)
```

Результат виконання:

```
Розподіл класів до обробки: Counter({np.float64(1.0): 1250, np.float64(0.0): 250})
Розподіл класів після SMOTE: Counter({np.float64(1.0): 871, np.float64(0.0): 871})

Classification Report:
              precision    recall  f1-score   support

         0.0       0.43      0.66      0.52        71
         1.0       0.93      0.84      0.88       379

    accuracy                           0.81       450
   macro avg       0.68      0.75      0.70       450
weighted avg       0.85      0.81      0.82       450


Confusion Matrix:
[[ 47  24]
 [ 62 317]]
```

Візуалізація:



Завдання 2.3. Знаходження оптимальних навчальних параметрів за допомогою сіткового пошуку

Використовуючи дані, що містяться у файлі data_random_forests.txt знайти оптимальних навчальних параметрів за допомогою сіткового пошуку.

Лістинг коду:

```
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# 1. Завантаження даних
try:
    data = np.loadtxt('data_random_forests.txt', delimiter=",")
except FileNotFoundError:
    raise FileNotFoundError("Файл 'data_random_forests.txt' не знайдено. Переко-
найтеся, що файл існує.")

# Розбиття даних на ознаки (X) та мітки (y)
X, y = data[:, :-1], data[:, -1]

# 2. Розділення на тренувальні та тестові дані
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ran-
dom_state=42)

# 3. Визначення параметрів для сіткового пошуку
param_grid = {
    'n_estimators': [50, 100, 200],  # Кількість дерев у лісі
    'max_depth': [None, 10, 20, 30],  # Максимальна глибина дерева
```

```
    'min_samples_split': [2, 5, 10],  # Мінімальна кількість зразків для поділу вузла
    'min_samples_leaf': [1, 2, 4],  # Мінімальна кількість зразків у листі
    'bootstrap': [True, False]  # Використання підвибірки (Bootstrap)
}

# 4. Ініціалізація класифікатора
rf_clf = RandomForestClassifier(random_state=42)

# 5. Сітковий пошук із перехресною перевіркою
grid_search = GridSearchCV(
    estimator=rf_clf,
    param_grid=param_grid,
    cv=5,  # Кількість фолдів для крос-валідації
    scoring='accuracy',  # Метрика для оцінки
    verbose=2,  # Рівень деталізації логів
    n_jobs=-1  # Використання всіх процесорів
)

# Навчання на тренувальних даних
grid_search.fit(X_train, y_train)

# 6. Виведення результатів
print("\nНайкращі параметри моделі:")
print(grid_search.best_params_)

print("\nНайкраща точність на тренувальних даних:")
print(f"{grid_search.best_score_:.4f}")

# Оцінка моделі з найкращими параметрами на тестових даних
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

print("\nClassification Report на тестових даних:")
print(classification_report(y_test, y_pred))
```

Результат виконання:

```
"C:\Program Files\Python313\python.exe" "D:\Лаби\4 КУРС\Системи штучного інтелекту\lab5\LR_5_task_3.py"
Fitting 5 folds for each of 216 candidates, totalling 1080 fits
[CV] END bootstrap=True, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=50; total time=   0.0s
[CV] END bootstrap=True, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=50; total time=   0.1s
[CV] END bootstrap=True, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=50; total time=   0.1s
[CV] END bootstrap=True, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=50; total time=   0.0s
[CV] END bootstrap=True, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=50; total time=   0.0s
[CV] END bootstrap=True, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   0.2s
[CV] END bootstrap=True, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   0.2s
[CV] END bootstrap=True, max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=50; total time=   0.0s
[CV] END bootstrap=True, max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=50; total time=   0.0s
[CV] END bootstrap=True, max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=50; total time=   0.0s
[CV] END bootstrap=True, max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=50; total time=   0.0s
[CV] END bootstrap=True, max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=50; total time=   0.0s
[CV] END bootstrap=True, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   0.3s
[CV] END bootstrap=True, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   0.3s
[CV] END bootstrap=True, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   0.3s
[CV] END bootstrap=True, max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=100; total time=   0.2s
[CV] END bootstrap=True, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=   0.5s
[CV] END bootstrap=True, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=   0.5s
[CV] END bootstrap=True, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=   0.5s
[CV] END bootstrap=True, max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=100; total time=   0.2s
[CV] END bootstrap=True, max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=100; total time=   0.2s
```

```
Найкращі параметри моделі:
{'bootstrap': True, 'max_depth': None, 'min_samples_leaf': 4, 'min_samples_split': 10, 'n_estimators': 50}

Найкраща точність на тренувальних даних:
0.8286

Classification Report на тестових даних:
              precision    recall  f1-score   support

         0.0       0.87      0.88      0.87        98
         1.0       0.82      0.82      0.82        83
         2.0       0.88      0.87      0.87        89

    accuracy                           0.86       270
   macro avg       0.85      0.85      0.85       270
weighted avg       0.86      0.86      0.86       270


Process finished with exit code 0
```

Завдання 2.5. Прогнозування інтенсивності дорожнього руху за допомогою класифікатора на основі гранично випадкових лісів.

Лістинг коду:

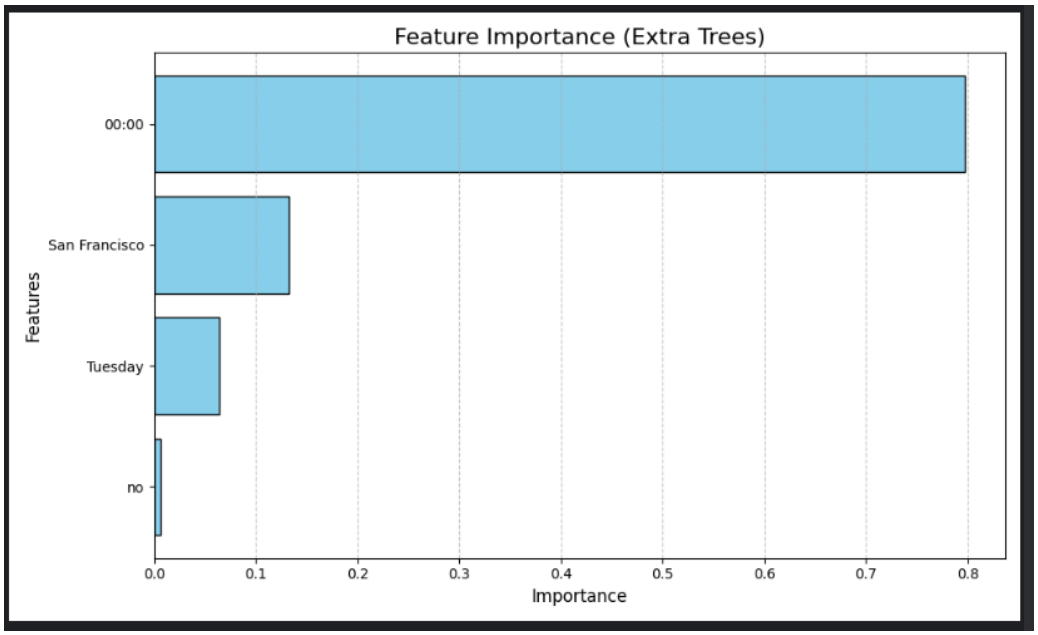
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import classification_report
from imblearn.over_sampling import RandomOverSampler

# 1. Завантаження даних із текстового файлу
try:
    data = pd.read_csv('traffic_data.txt', delimiter=",")
except FileNotFoundError:
    raise FileNotFoundError("Файл 'traffic_data.txt' не знайдено. Перевірте шлях до файлу.")

# 2. Перетворення текстових даних у числові
label_encoders = {}
for column in data.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    data[column] = le.fit_transform(data[column])
    label_encoders[column] = le  # Збереження енкодера для подальшого декодування

# 3. Відділення ознак і цільової змінної
X = data.iloc[:, :-1]  # Усі стовпці, крім останнього
y = data.iloc[:, -1]   # Останній стовпець — цільова змінна

# Перевірка та балансування класів
ros = RandomOverSampler(random_state=42)
X_resampled, y_resampled = ros.fit_resample(X, y)

# Масштабування ознак (опціонально)
scaler = StandardScaler()
X_resampled = scaler.fit_transform(X_resampled)
```

```
# 4. Розділення на тренувальні та тестові дані
X_train, X_test, y_train, y_test = train_test_split(
    X_resampled, y_resampled, test_size=0.3, random_state=42, stratify=y_resampled
)

# 5. Створення та навчання класифікатора Extra Trees
et_clf = ExtraTreesClassifier(n_estimators=100, random_state=42)
et_clf.fit(X_train, y_train)

# 6. Прогнозування та оцінка
y_pred = et_clf.predict(X_test)
print("Класифікаційний звіт:")
print(classification_report(y_test, y_pred, zero_division=1))

# 7. Важливість ознак
feature_importances = et_clf.feature_importances_
feature_importance_data = {
    "Feature": data.columns[:-1],
    "Importance": feature_importances
}
importance_df = pd.DataFrame(feature_importance_data).sort_values(by="Importance",
ascending=False)

print("\nВажливість ознак:")
print(importance_df.to_string(index=False))

# 8. Графік важливості ознак
def plot_feature_importance(importance_df):
    """
    Побудова графіка важливості ознак.
    """
    plt.figure(figsize=(10, 6))
    plt.title("Feature Importance (Extra Trees)", fontsize=16)
    plt.barh(importance_df["Feature"], importance_df["Importance"], col-
or="skyblue", edgecolor="black")
    plt.xlabel("Importance", fontsize=12)
    plt.ylabel("Features", fontsize=12)
    plt.gca().invert_yaxis()  # Перевертає осі для кращого вигляду
    plt.grid(axis='x', linestyle='--', alpha=0.7)
    plt.tight_layout()
    plt.show()

plot_feature_importance(importance_df)
```

Результат виконання:

```
          54       0.95      1.00      0.98       274
          55       0.99      1.00      0.99       274
          56       1.00      1.00      1.00       273
          57       1.00      1.00      1.00       273
          58       1.00      1.00      1.00       274
          59       1.00      1.00      1.00       274
          60       1.00      1.00      1.00       274
          61       0.98      1.00      0.99       273
          62       1.00      1.00      1.00       274
          63       1.00      1.00      1.00       274
          64       1.00      1.00      1.00       274
          66       1.00      1.00      1.00       274
          70       1.00      1.00      1.00       273
          75       1.00      1.00      1.00       274
          90       1.00      1.00      1.00       274

    accuracy                           0.84     19152
   macro avg       0.84      0.84      0.84     19152
weighted avg       0.84      0.84      0.84     19152


Важливість ознак:
      Feature  Importance
        00:00    0.797157
San Francisco    0.132222
      Tuesday    0.063868
           no    0.006754

Process finished with exit code 0
```

Візуалізація:



Висновок: використовуючи спеціалізовані бібліотеки та мову програмування Python дослідив методи ансамблів у машинному навчанні.