Практичне завдання: Бінарна класифікація на основі даних Titanic
Мета:
Закріпити знання, отримані під час лекції з класифікації, застосовуючи різні моделі машинного навчання для прогнозування виживання пасажирів на основі набору даних Titanic. Практичне завдання спрямоване на розвиток навичок роботи з реальними даними, підготовки даних, побудови моделей та їх оцінки.

Встановіть необхідні бібліотеки:

```
pip install pandas numpy scikit-learn matplotlib seaborn kaggle
```

Завантажте дані Titanic з Kaggle або використайте вбудовані дані:

```
import seaborn as sns
import pandas as pd

# Використання вбудованих даних Titanic з бібліотеки seaborn
data = sns.load dataset('titanic')
```

Виведіть перші 10 рядків , отримати базову статистику і дослідіть пропущені значення:

```
print(data.head(10))
print(data.describe())
print(data.isnull().sum())
```

Обробка пропущених значень , закодуйте категоріальні змінні , створити нові ознаки:

```
data['age'].fillna(data['age'].mean(), inplace=True)
data['embarked'].fillna(data['embarked'].mode()[0], inplace=True)
data = pd.get_dummies(data, columns=['sex', 'embarked'], drop_first=True)
data['family_size'] = data['sibsp'] + data['parch']
```

Поділ даних на тренувальну та тестову вибірки

```
from sklearn.model_selection import train_test_split

X = data.drop(['survived'], axis=1)
y = data['survived']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

Побудова моделей

1. Логістична регресія
2. Дерева рішень
3. Випадкові ліси:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_sc

log_reg = LogisticRegression(max_iter=200)
log_reg.fit(X_train, y_train)
y_pred_log = log_reg.predict(X_test)

# Оцінка метрик
print(f'Accuracy: {accuracy_score(y_test, y_pred_log)}')
print(f'Precision: {precision_score(y_test, y_pred_log)}')
print(f'Recall: {recall_score(y_test, y_pred_log)}')
print(f'F1-Score: {f1_score(y_test, y_pred_log)}')
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
y_pred_dt = dt.predict(X_test)

# Оцінка метрик
print(f'Accuracy: {accuracy_score(y_test, y_pred_dt)}')
print(f'Precision: {precision_score(y_test, y_pred_dt)}')
print(f'Recall: {recall_score(y_test, y_pred_dt)}')
print(f'F1-Score: {f1_score(y_test, y_pred_dt)}')
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)

# Оцінка метрик
print(f'Accuracy: {accuracy_score(y_test, y_pred_rf)}')
print(f'Precision: {precision_score(y_test, y_pred_rf)}')
print(f'Recall: {recall_score(y_test, y_pred_rf)}')
print(f'F1-Score: {f1_score(y_test, y_pred_rf)}')
```

Оцінка результатів

1. Матриця плутанини
2. ROC-крива та AUC

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

cm_log = confusion_matrix(y_test, y_pred_log)
sns.heatmap(cm_log, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix - Logistic Regression')
plt.show()
from sklearn.metrics import roc_curve, auc

fpr_log, tpr_log, _ = roc_curve(y_test, log_reg.predict_proba(X_test)[:,1])
roc_auc_log = auc(fpr_log, tpr_log)

plt.figure()
plt.plot(fpr_log, tpr_log, color='blue', lw=2, label='Logistic Regression (AUC =
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```

Оптимізація моделі

1. Крос-валідація
2. Оптимізація гіперпараметрів для випадкового лісу:

```
from sklearn.model_selection import cross_val_score

log_reg_cv = cross_val_score(log_reg, X, y, cv=5)
dt_cv = cross_val_score(dt, X, y, cv=5)

print(f'Logistic Regression CV: {log_reg_cv.mean()}')
print(f'Decision Tree CV: {dt_cv.mean()}')
from sklearn.model_selection import GridSearchCV

param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30]
}

grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5)
grid_search.fit(X_train, y_train)
print(f'Best Parameters: {grid search.best params }')
```

Завдання з творчим підходом

1. Проаналізуйте важливість ознак для моделі випадкового лісу:

```
importances = rf.feature_importances_
feature_names = X.columns
feature_importances = pd.Series(importances, index=feature_names).sort_values(ascending=False)
print(feature_importances)
```

Висновок

У процесі виконання завдання ми провели комплексний аналіз даних на прикладі набору даних Titanic. Було виконано передобробку даних, включаючи обробку пропущених значень та кодування категоріальних змінних. Ми побудували декілька моделей класифікації, таких як логістична регресія, дерева рішень та випадкові ліси, і оцінили їх за допомогою різних метрик. Крім того, були проведені крос-валідація та оптимізація моделей, що дозволило підвищити їх точність і надійність. Результати роботи були опубліковані на GitHub, що забезпечує доступність та можливість подальшого аналізу. Загалом, це завдання допомогло набути практичних навичок у машинному навчанні та обробці даних, що є важливими для роботи у сфері Data Science.