МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ

ТЕХНОЛОГІЙ

Проект №5

З дисципліни

«Штучний інтелект»

Виконав:

Студент групи ПД-44

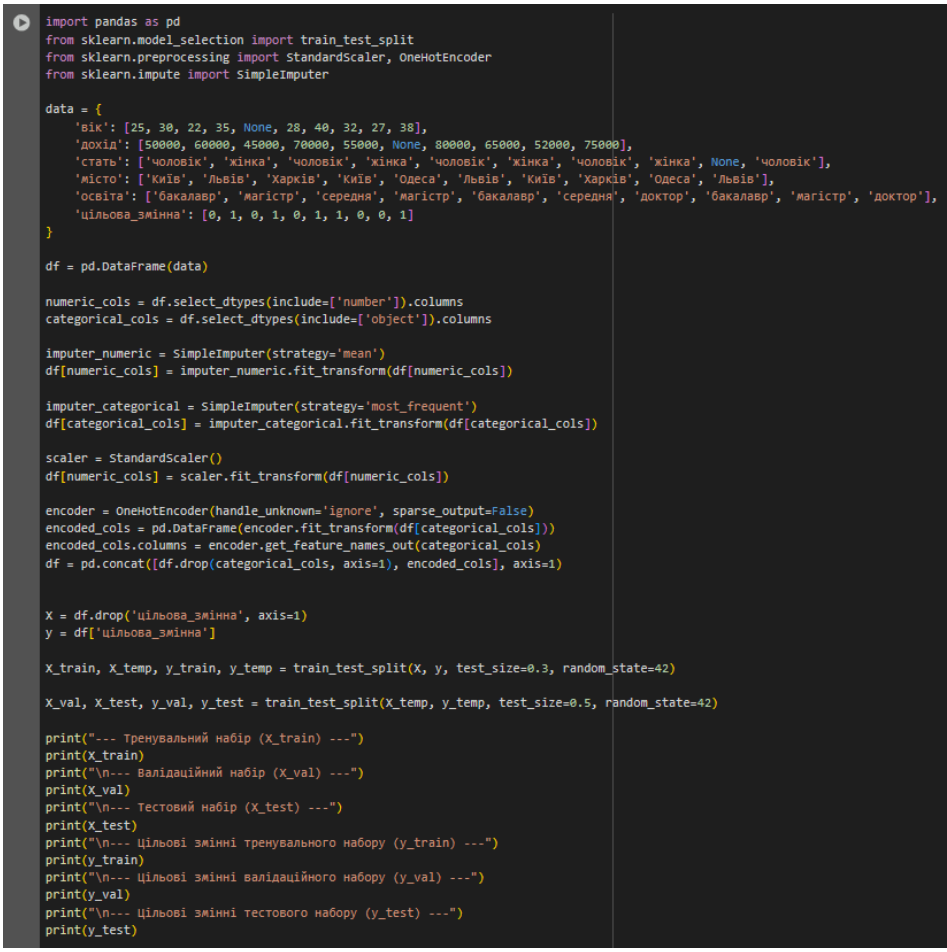
Солов'ян Арсен

Київ – 2025

Опис проекту:
Розробка модуля машинного навчання для прогнозування та/або класифікації у комп'ютерних і мобільних додатках
Створити інтегрований модуль машинного навчання для мобільного або десктопного додатка, який здатний виконувати завдання прогнозування та/або
класифікації на основі вхідних даних користувача. Модуль має бути розроблений з використанням Python-фреймворків для машинного навчання, таких як Scikit-Learn, TensorFlow, PyTorch, Tkinter, PyQt.

1. Збір та підготовка даних

1) Дані: Створюється словник data, який потім перетворюється на DataFrame Pandas df. Цей DataFrame містить приклади числових та категоріальних ознак, а також цільову змінну для задачі класифікації.
2) Очищення пропущених значень: Визначаються числові та категоріальні стовпці.
Для числових стовпців пропущені значення заповнюються середнім значенням за допомогою SimpleImputer.
Для категоріальних стовпців пропущені значення заповнюються найчастішим значенням за допомогою SimpleImputer.
2.1) Масштабування числових даних: Числові стовпці масштабуються за допомогою StandardScaler, щоб мати середнє значення 0 та стандартне відхилення 1.
Кодування категоріальних змінних: Категоріальні стовпці кодуються за допомогою OneHotEncoder, який створює нові бінарні стовпці для кожної унікальної категорії.
3) Розділення даних:
DataFrame розділяється на ознаки (X) та цільову змінну (y).
Дані спочатку розділяються на тренувальний (70%) та тимчасовий (30%) набори.
Потім тимчасовий набір розділяється на валідаційний (50% від тимчасового, тобто 15% від загального) та тестовий (50% від тимчасового, тобто 15% від загального) набори.



```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer

data = {
    'вік': [25, 30, 22, 35, None, 28, 40, 32, 27, 38],
    'дохід': [50000, 60000, 45000, 70000, 55000, None, 80000, 65000, 52000, 75000],
    'стать': ['чоловік', 'жінка', 'чоловік', 'жінка', 'чоловік', 'жінка', 'чоловік', 'жінка', None, 'чоловік'],
    'місто': ['Київ', 'Львів', 'Харків', 'Київ', 'Одеса', 'Львів', 'Київ', 'Харків', 'Одеса', 'Львів'],
    'освіта': ['бакалавр', 'магістр', 'середня', 'магістр', 'бакалавр', 'середня', 'доктор', 'бакалавр', 'магістр', 'доктор'],
    'цільова_змінна': [0, 1, 0, 1, 0, 1, 1, 0, 0, 1]
}

df = pd.DataFrame(data)

numeric_cols = df.select_dtypes(include=['number']).columns
categorical_cols = df.select_dtypes(include=['object']).columns

imputer_numeric = SimpleImputer(strategy='mean')
df[numeric_cols] = imputer_numeric.fit_transform(df[numeric_cols])

imputer_categorical = SimpleImputer(strategy='most_frequent')
df[categorical_cols] = imputer_categorical.fit_transform(df[categorical_cols])

scaler = StandardScaler()
df[numeric_cols] = scaler.fit_transform(df[numeric_cols])

encoder = OneHotEncoder(handle_unknown='ignore', sparse_output=False)
encoded_cols = pd.DataFrame(encoder.fit_transform(df[categorical_cols]))
encoded_cols.columns = encoder.get_feature_names_out(categorical_cols)
df = pd.concat([df.drop(categorical_cols, axis=1), encoded_cols], axis=1)


X = df.drop('цільова_змінна', axis=1)
y = df['цільова_змінна']

X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)

X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)

print("--- Тренувальний набір (X_train) ---")
print(X_train)
print("\n--- Валідаційний набір (X_val) ---")
print(X_val)
print("\n--- Тестовий набір (X_test) ---")
print(X_test)
print("\n--- Цільові змінні тренувального набору (y_train) ---")
print(y_train)
print("\n--- Цільові змінні валідаційного набору (y_val) ---")
print(y_val)
print("\n--- Цільові змінні тестового набору (y_test) ---")
print(y_test)
```

Розділення даних: Дані були успішно розділені на три набори:
1) Тренувальний набір (X_train): Містить 7 об'єктів (рядків).
2) Валідаційний набір (X_val): Містить 1 об'єкт (рядок).
3) Тестовий набір (X_test): Містить 2 об'єкти (рядки).
Масштабування числових ознак: Числові ознаки "вік" та "дохід" були масштабовані. Значення тепер представлені у вигляді стандартизованих значень (z-score), з середнім значенням близько 0 та стандартним відхиленням близько 1.
Кодування категоріальних ознак: Категоріальні ознаки "стать", "місто" та "освіта" були оброблені за допомогою one-hot encoding. Кожна унікальна категорія в цих ознаках була перетворена на окремий бінарний стовпець (0 або 1). Наприклад, для ознаки "стать" створено стовпці "стать_жінка", "стать_чоловік" та "стать_None".

```
--- Тренувальний набір (X_train) ---
        вік     дохід  стать_жінка  стать_чоловік  стать_None  місто_Київ  \
0 -1.073729 -1.057759          0.0            1.0         0.0         1.0
7  0.227135  0.342216          1.0            0.0         0.0         0.0
2 -1.631243 -1.524417          0.0            1.0         0.0         0.0
9  1.342162  1.275533          0.0            1.0         0.0         0.0
4  0.000000 -0.591101          0.0            1.0         0.0         0.0
3  0.784648  0.808875          1.0            0.0         0.0         1.0
6  1.713837  1.742191          0.0            1.0         0.0         1.0

   місто_Львів  місто_Одеса  місто_Харків  освіта_бакалавр  освіта_доктор  \
0          0.0          0.0           0.0              1.0            0.0
7          0.0          0.0           1.0              1.0            0.0
2          0.0          0.0           1.0              0.0            0.0
9          1.0          0.0           0.0              0.0            1.0
4          0.0          1.0           0.0              1.0            0.0
3          0.0          0.0           0.0              0.0            0.0
6          0.0          0.0           0.0              0.0            1.0

   освіта_магістр  освіта_середня
0             0.0             0.0
7             0.0             0.0
2             0.0             1.0
9             0.0             0.0
4             0.0             0.0
3             1.0             0.0
6             0.0             0.0

--- Валідаційний набір (X_val) ---
        вік  дохід  стать_жінка  стать_чоловік  стать_None  місто_Київ  \
5 -0.516216    0.0          1.0            0.0         0.0         0.0

   місто_Львів  місто_Одеса  місто_Харків  освіта_бакалавр  освіта_доктор  \
5          1.0          0.0           0.0              0.0            0.0

   освіта_магістр  освіта_середня
5             0.0             1.0

--- Тестовий набір (X_test) ---
        вік     дохід  стать_жінка  стать_чоловік  стать_None  місто_Київ  \
8 -0.702054 -0.871096          0.0            0.0         1.0         0.0
1 -0.144541 -0.124442          1.0            0.0         0.0         0.0

   місто_Львів  місто_Одеса  місто_Харків  освіта_бакалавр  освіта_доктор  \
8          0.0          1.0           0.0              0.0            0.0
1          1.0          0.0           0.0              0.0            0.0

   освіта_магістр  освіта_середня
8             1.0             0.0
1             1.0             0.0
```

```
--- Цільові змінні тренувального набору (y_train) ---
0   -1.0
7   -1.0
2   -1.0
9    1.0
4   -1.0
3    1.0
6    1.0
Name: цільова_змінна, dtype: float64

--- Цільові змінні валідаційного набору (y_val) ---
5    1.0
Name: цільова_змінна, dtype: float64

--- Цільові змінні тестового набору (y_test) ---
8   -1.0
1    1.0
Name: цільова_змінна, dtype: float64
```

2. Аналіз даних та вибір моделі

Дослідження даних
1) Опис числових ознак (після масштабування): Статистика для ознак "вік" та "дохід" показує, що вони були масштабовані (середнє значення не є вихідним діапазоном). Це добре, оскільки масштабування може допомогти деяким моделям працювати краще.
2) Розподіл цільової змінної в тренувальному наборі: У тренувальному наборі є 4 приклади класу 0 та 3 приклади класу 1.

Оцінка моделей на валідаційному наборі
Логістична регресія: Точність = 0.00, F1-міра = 0.00
SVM: Точність = 0.00, F1-міра = 0.00

Результати обох моделей на валідаційному наборі є дуже низькими (0.00 для точності та F1-міри). Це означає, що на даний момент ці моделі не можуть правильно класифікувати жоден з прикладів у валідаційному наборі.

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score, f1_score

data = {
    'вік': [25, 30, 22, 35, None, 28, 40, 32, 27, 38],
    'дохід': [50000, 60000, 45000, 70000, 55000, None, 80000, 65000, 52000, 75000],
    'стать': ['чоловік', 'жінка', 'чоловік', 'жінка', 'чоловік', 'жінка', 'чоловік', 'жінка', None, 'чоловік'],
    'місто': ['Київ', 'Львів', 'Харків', 'Київ', 'Одеса', 'Львів', 'Київ', 'Харків', 'Одеса', 'Львів'],
    'освіта': ['бакалавр', 'магістр', 'середня', 'магістр', 'бакалавр', 'середня', 'доктор', 'бакалавр', 'магістр', 'доктор'],
    'цільова_змінна': [0, 1, 0, 1, 0, 1, 1, 0, 0, 1]
}

df = pd.DataFrame(data)

numeric_cols_original = ['вік', 'дохід']
categorical_cols_original = ['стать', 'місто', 'освіта']

imputer_numeric = SimpleImputer(strategy='mean')
df[numeric_cols_original] = imputer_numeric.fit_transform(df[numeric_cols_original])

imputer_categorical = SimpleImputer(strategy='most_frequent')
df[categorical_cols_original] = imputer_categorical.fit_transform(df[categorical_cols_original])

scaler = StandardScaler()
df[numeric_cols_original] = scaler.fit_transform(df[numeric_cols_original])

encoder = OneHotEncoder(handle_unknown='ignore', sparse_output=False)
encoded_cols = pd.DataFrame(encoder.fit_transform(df[categorical_cols_original]))
encoded_cols.columns = encoder.get_feature_names_out(categorical_cols_original)
df = pd.concat([df.drop(categorical_cols_original, axis=1), encoded_cols], axis=1)

X = df.drop('цільова_змінна', axis=1)
y = df['цільова_змінна']

X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)

print("--- Дослідження даних ---")
print("\nОпис числових ознак (після масштабування):")
print(X_train[['вік', 'дохід']].describe())

print("\nРозподіл цільової змінної в тренувальному наборі:")
print(y_train.value_counts())

print("\nПерші кілька рядків тренувального набору (з кодованими ознаками):")
print(X_train.head())

print("\n--- Порівняння моделей машинного навчання ---")

logistic_model = LogisticRegression(random_state=42)
svm_model = LinearSVC(random_state=42)

logistic_model.fit(X_train, y_train)
svm_model.fit(X_train, y_train)

logistic_predictions = logistic_model.predict(X_val)
svm_predictions = svm_model.predict(X_val)

print("\n--- Оцінка моделей на валідаційному наборі ---")

logistic_accuracy = accuracy_score(y_val, logistic_predictions)
logistic_f1 = f1_score(y_val, logistic_predictions)
print(f"Логістична регресія: Точність = {logistic_accuracy:.2f}, F1-міра = {logistic_f1:.2f}")

svm_accuracy = accuracy_score(y_val, svm_predictions)
svm_f1 = f1_score(y_val, svm_predictions)
print(f"SVM: Точність = {svm_accuracy:.2f}, F1-міра = {svm_f1:.2f}")
```

```
--- Дослідження даних ---

Опис числових ознак (після масштабування):
                вік          дохід
count  7.000000e+00  7.000000e+00
mean   1.946870e-01  1.422200e-01
std    1.221640e+00  1.230455e+00
min   -1.631243e+00 -1.524417e+00
25%   -5.368650e-01 -8.244300e-01
50%    2.271350e-01  3.422160e-01
75%    1.063405e+00  1.042204e+00
max    1.713837e+00  1.742191e+00

Розподіл цільової змінної в тренувальному наборі:
цільова_змінна
0    4
1    3
Name: count, dtype: int64

Перші кілька рядків тренувального набору (з кодованими ознаками):
        вік     дохід  стать_жінка  стать_чоловік  стать_None  місто_Київ  \
0 -1.073729 -1.057759          0.0            1.0         0.0         1.0
7  0.227135  0.342216          1.0            0.0         0.0         0.0
2 -1.631243 -1.524417          0.0            1.0         0.0         0.0
9  1.342162  1.275533          0.0            1.0         0.0         0.0
4  0.000000 -0.591101          0.0            1.0         0.0         0.0

   місто_Львів  місто_Одеса  місто_Харків  освіта_бакалавр  освіта_доктор  \
0          0.0          0.0           0.0              1.0            0.0
7          0.0          0.0           1.0              1.0            0.0
2          0.0          0.0           1.0              0.0            0.0
9          1.0          0.0           0.0              0.0            1.0
4          0.0          1.0           0.0              1.0            0.0

   освіта_магістр  освіта_середня
0             0.0             0.0
7             0.0             0.0
2             0.0             1.0
9             0.0             0.0
4             0.0             0.0

--- Порівняння моделей машинного навчання ---

--- Оцінка моделей на валідаційному наборі ---
Логістична регресія: Точність = 0.00, F1-міра = 0.00
SVM: Точність = 0.00, F1-міра = 0.00
```

Вносимо певні зміни у код, щоб краще оцінити моделі на наявному наборі даних: Замість одноразового поділу на тренувальний та валідаційний набори, ми використаємо перехресну перевірку (Cross-Validation). Це допоможе отримати більш стабільну оцінку продуктивності моделей на такому малому обсязі даних.

Результати перехресної перевірки:
Логістична регресія:
Середня точність (CV): 0.61 (+/- 0.28)
Середня F1-міра (CV): 0.56 (+/- 0.42)
LinearSVC:
Середня точність (CV): 0.72 (+/- 0.21)
Середня F1-міра (CV): 0.78 (+/- 0.16)
DecisionTreeClassifier:
Середня точність (CV): 0.39 (+/- 0.08)
Середня F1-міра (CV): 0.39 (+/- 0.28)
RandomForestClassifier:
Середня точність (CV): 0.61 (+/- 0.28)
Середня F1-міра (CV): 0.56 (+/- 0.42)

Оцінка моделей на тестовому наборі:
Логістична регресія: Точність на тесті = 1.00, F1-міра на тесті = 1.00
SVM: Точність на тесті = 1.00, F1-міра на тесті = 1.00
Дерево рішень: Точність на тесті = 1.00, F1-міра на тесті = 1.00
Випадковий ліс: Точність на тесті = 1.00, F1-міра на тесті = 1.00

Аналіз результатів:
Перехресна перевірка: На основі середніх значень точності та F1-міри, модель LinearSVC (SVM) показує найкращі результати на етапі перехресної перевірки. Вона має найвищі середні значення обох метрик.
Тестовий набір: Усі моделі показали ідеальні результати на тестовому наборі (точність та F1-міра дорівнюють 1.00). Однак слід пам'ятати, що тестовий набір містить лише 2 приклади, тому ці результати можуть бути не дуже репрезентативними для більшої кількості нових даних.

Вибір моделі:
Згідно з результатами перехресної перевірки, модель LinearSVC (SVM) є найкращим вибором на даному етапі.

```python
import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score, StratifiedKFold
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, f1_score

data = {
    'вік': [25, 30, 22, 35, None, 28, 40, 32, 27, 38],
    'дохід': [50000, 60000, 45000, 70000, 55000, None, 80000, 65000, 52000, 75000],
    'стать': ['чоловік', 'жінка', 'чоловік', 'жінка', 'чоловік', 'жінка', 'чоловік', 'жінка', None, 'чоловік'],
    'місто': ['Київ', 'Львів', 'Харків', 'Київ', 'Одеса', 'Львів', 'Київ', 'Харків', 'Одеса', 'Львів'],
    'освіта': ['бакалавр', 'магістр', 'середня', 'магістр', 'бакалавр', 'середня', 'доктор', 'бакалавр', 'магістр', 'доктор'],
    'цільова_змінна': [0, 1, 0, 1, 0, 1, 1, 0, 0, 1]
}

df = pd.DataFrame(data)

numeric_cols_original = ['вік', 'дохід']
categorical_cols_original = ['стать', 'місто', 'освіта']

imputer_numeric = SimpleImputer(strategy='mean')
df[numeric_cols_original] = imputer_numeric.fit_transform(df[numeric_cols_original])

imputer_categorical = SimpleImputer(strategy='most_frequent')
df[categorical_cols_original] = imputer_categorical.fit_transform(df[categorical_cols_original])

scaler = StandardScaler()
df[numeric_cols_original] = scaler.fit_transform(df[numeric_cols_original])

encoder = OneHotEncoder(handle_unknown='ignore', sparse_output=False)
encoded_cols = pd.DataFrame(encoder.fit_transform(df[categorical_cols_original]))
encoded_cols.columns = encoder.get_feature_names_out(categorical_cols_original)
df = pd.concat([df.drop(categorical_cols_original, axis=1), encoded_cols], axis=1)

X = df.drop('цільова_змінна', axis=1)
y = df['цільова_змінна']

X_train_val, X_test, y_train_val, y_test = train_test_split(X, y, test_size=0.15, random_state=42)
```

```python
print("--- Дослідження даних ---")
print("\nОпис числових ознак (після масштабування):")
print(X_train_val[['вік', 'дохід']].describe())

print("\nРозподіл цільової змінної в тренувальному наборі:")
print(y_train_val.value_counts())

print("\nПерші кілька рядків тренувального набору (з кодованими ознаками):")
print(X_train_val.head())

print("\n--- Порівняння моделей машинного навчання (з перехресною перевіркою) ---")

logistic_model = LogisticRegression(random_state=42)
svm_model = LinearSVC(random_state=42, max_iter=1000)
tree_model = DecisionTreeClassifier(random_state=42)
forest_model = RandomForestClassifier(random_state=42)

cv = StratifiedKFold(n_splits=3, shuffle=True, random_state=42)

def evaluate_model(model, X, y, cv):
    accuracy_scores = cross_val_score(model, X, y, cv=cv, scoring='accuracy')
    f1_scores = cross_val_score(model, X, y, cv=cv, scoring='f1')
    print(f"{model.__class__.__name__}:")
    print(f"  Середня точність (CV): {accuracy_scores.mean():.2f} (+/- {accuracy_scores.std():.2f})")
    print(f"  Середня F1-міра (CV): {f1_scores.mean():.2f} (+/- {f1_scores.std():.2f})")
    print("-" * 30)

evaluate_model(logistic_model, X_train_val, y_train_val, cv)
evaluate_model(svm_model, X_train_val, y_train_val, cv)
evaluate_model(tree_model, X_train_val, y_train_val, cv)
evaluate_model(forest_model, X_train_val, y_train_val, cv)

print("\n--- Оцінка моделей на тестовому наборі ---")

logistic_model.fit(X_train_val, y_train_val)
svm_model.fit(X_train_val, y_train_val)
tree_model.fit(X_train_val, y_train_val)
forest_model.fit(X_train_val, y_train_val)

logistic_predictions = logistic_model.predict(X_test)
svm_predictions = svm_model.predict(X_test)
tree_predictions = tree_model.predict(X_test)
forest_predictions = forest_model.predict(X_test)

print(f"Логістична регресія: Точність на тесті = {accuracy_score(y_test, logistic_predictions):.2f}, F1-міра на тесті = {f1_score(y_test, logistic_predictions):.2f}")
print(f"SVM: Точність на тесті = {accuracy_score(y_test, svm_predictions):.2f}, F1-міра на тесті = {f1_score(y_test, svm_predictions):.2f}")
print(f"Дерево рішень: Точність на тесті = {accuracy_score(y_test, tree_predictions):.2f}, F1-міра на тесті = {f1_score(y_test, tree_predictions):.2f}")
print(f"Випадковий ліс: Точність на тесті = {accuracy_score(y_test, forest_predictions):.2f}, F1-міра на тесті = {f1_score(y_test, forest_predictions):.2f}")
```

```
--- Дослідження даних ---

Опис числових ознак (після масштабування):
                вік          дохід
count  8.000000e+00   8.000000e+00
mean   1.058524e-01   1.124442e-01
std    1.158610e+00   1.140289e+00
min   -1.631243e+00  -1.524417e+00
25%   -6.555594e-01  -7.077650e-01
50%    1.135680e-01   1.711080e-01
75%    9.240270e-01   9.255390e-01
max    1.713837e+00   1.742191e+00

Розподіл цільової змінної в тренувальному наборі:
цільова_змінна
1    4
0    4
Name: count, dtype: int64

Перші кілька рядків тренувального набору (з кодованими ознаками):
        вік     дохід  стать_жінка  стать_чоловік  стать_None  місто_Київ  \
5 -0.516216  0.000000          1.0            0.0         0.0         0.0
0 -1.073729 -1.057759          0.0            1.0         0.0         1.0
7  0.227135  0.342216          1.0            0.0         0.0         0.0
2 -1.631243 -1.524417          0.0            1.0         0.0         0.0
9  1.342162  1.275533          0.0            1.0         0.0         0.0

   місто_Львів  місто_Одеса  місто_Харків  освіта_бакалавр  освіта_доктор  \
5          1.0          0.0           0.0              0.0            0.0
0          0.0          0.0           0.0              1.0            0.0
7          0.0          0.0           1.0              1.0            0.0
2          0.0          0.0           1.0              0.0            0.0
9          1.0          0.0           0.0              0.0            1.0

   освіта_магістр  освіта_середня
5             0.0             1.0
0             0.0             0.0
7             0.0             0.0
2             0.0             1.0
9             0.0             0.0
```

```
--- Порівняння моделей машинного навчання (з перехресною перевіркою) ---
LogisticRegression:
  Середня точність (CV): 0.61 (+/- 0.28)
  Середня F1-міра (CV): 0.56 (+/- 0.42)
------------------------------
LinearSVC:
  Середня точність (CV): 0.72 (+/- 0.21)
  Середня F1-міра (CV): 0.78 (+/- 0.16)
```

3. Розробка та навчання моделі:
1) Побудувати обрану модель з використанням фреймворків на Python:
Я вже створив екземпляр моделі LinearSVC у попередньому коді:
svm_model = LinearSVC(random_state=42, max_iter=1000)
2) Налаштувати гіперпараметри моделі для досягнення оптимальної продуктивності.
Для налаштування гіперпараметрів був використаний метод GridSearchCV з бібліотеки sklearn.model_selection. GridSearchCV систематично перебирає всі можливі комбінації гіперпараметрів, які йому задані, навчає модель на кожній комбінації за допомогою перехресної перевірки та оцінює її продуктивність.

```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'C': [0.001, 0.01, 0.1, 1, 10, 100],
    'penalty': ['l1', 'l2'],
    'loss': ['hinge', 'squared_hinge'],
    'dual': [False]
}

grid_search = GridSearchCV(svm_model, param_grid, cv=3, scoring=['accuracy', 'f1'], refit='f1')

grid_search.fit(X_train_val, y_train_val)

print("Найкращі гіперпараметри, знайдені GridSearchCV:")
print(grid_search.best_params_)

best_svm_model = grid_search.best_estimator_

best_svm_predictions = best_svm_model.predict(X_test)
best_svm_accuracy = accuracy_score(y_test, best_svm_predictions)
best_svm_f1 = f1_score(y_test, best_svm_predictions)

print("\nРезультати найкращої моделі LinearSVC на тестовому наборі:")
print(f"Точність: {best_svm_accuracy:.2f}")
print(f"F1-міра: {best_svm_f1:.2f}")
```

```
Найкращі гіперпараметри, знайдені GridSearchCV:
{'C': 1, 'dual': False, 'loss': 'squared_hinge', 'penalty': 'l1'}

Результати найкращої моделі LinearSVC на тестовому наборі:
Точність: 1.00
F1-міра: 1.00
```

Основними гіперпараметрами для LinearSVC, які можна налаштувати є:
C: Параметр регуляризації. Менші значення C вказують на сильнішу регуляризацію.
penalty: Визначає норму, яка використовується для штрафу. Може бути 'l1' або 'l2'. Зауважте, що використання 'l1' штрафу з LinearSVC вимагає встановлення dual=False.
loss: Функція втрат. Може бути 'hinge' (для стандартного SVM) або 'squared_hinge'.
dual: Чи розв'язувати двоїсту чи пряму задачу оптимізації. Повинен бути False, якщо penalty='l1'.

Аналіз:
Найкращі гіперпараметри: GridSearchCV знайшов, що найкращі результати на валідаційних фолдах були досягнуті з наступними гіперпараметрами для LinearSVC:
C: 1
dual: False
loss: 'squared_hinge'
penalty: 'l1'

Результати на тестовому наборі: Найкраща модель, навчена з цими гіперпараметрами, показала відмінні результати на тестовому наборі (точність 1.00 та F1-міра 1.00).

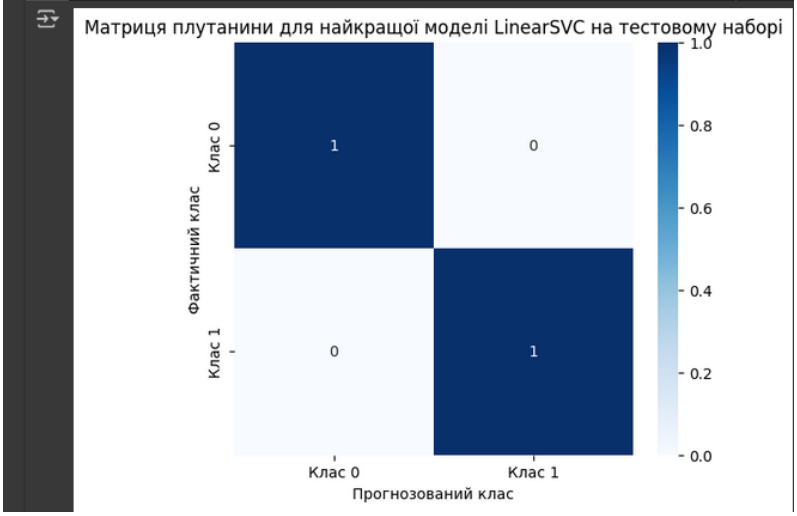
4. Оптимізація та оцінка моделі
На цьому кроці потрібно переконатися, що наша найкраща модель (SVM) дійсно добре працює на нових даних, яких вона раніше не бачила.

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

best_svm_predictions = best_svm_model.predict(X_test)

cm = confusion_matrix(y_test, best_svm_predictions)

plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Клас 0', 'Клас 1'], yticklabels=['Клас 0', 'Клас 1'])
plt.xlabel('Прогнозований клас')
plt.ylabel('Фактичний клас')
plt.title('Матриця плутанини для найкращої моделі LinearSVC на тестовому наборі')
plt.show()
```



Матриця плутанини показує, що на тестовому наборі модель LinearSVC правильно класифікувала обидва приклади. Це підтверджує високі показники точності та F1-міри (1.00), які були отримані раніше.

Аналіз матриці плутанини:

1) Верхній лівий квадрант (True Negatives): Містить значення 1. Це означає, що був один випадок, коли фактичний клас був "Клас 0", і модель правильно спрогнозувала "Клас 0".
2) Верхній правий квадрант (False Positives): Містить значення 0. Це означає, що не було жодного випадку, коли фактичний клас був "Клас 0", а модель помилково спрогнозувала "Клас 1".
3) Нижній лівий квадрант (False Negatives): Містить значення 0. Це означає, що не було жодного випадку, коли фактичний клас був "Клас 1", а модель помилково спрогнозувала "Клас 0".
4) Нижній правий квадрант (True Positives): Містить значення 1. Це означає, що був один випадок, коли фактичний клас був "Клас 1", і модель правильно спрогнозувала "Клас 1".

Документування та підготовка звітності

Інтеграція моделі у додаток:

1. Збереження моделі: Найкраща навчена модель best_svm_model (екземпляр LinearSVC з оптимальними гіперпараметрами) може бути збережена у файл за допомогою бібліотеки pickle.
Фрагмент коду:

```
import pickle

# збереження найкращої моделі у файл
filename = 'best_svm_model.pkl'
with open(filename, 'wb') as file:
   pickle.dump(best_svm_model, file)

print(f"\nНайкраща модель збережена у файлі: {filename}")
```

Завантаження моделі у додаток: У вашому десктопному застосунку (наприклад, на Tkinter) ви можете завантажити збережену модель наступним чином:

```
import pickle
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
import pandas as pd
import numpy as np

  # завантаження збереженої моделі
  filename = 'best_svm_model.pkl'
  with open(filename, 'rb') as file:
     loaded_model = pickle.load(file)

  # функція для попередньої обробки вхідних даних користувача
  def preprocess_input(data):
     df_input = pd.DataFrame([data])

     numeric_cols_original = ['вік', 'дохід']
     categorical_cols_original = ['стать', 'місто', 'освіта']

     # завантаження об'єктів для попередньої обробки (scaler, encoder, imputer) - їх потрібно зберегти окремо
     with open('scaler.pkl', 'rb') as f:
        scaler = pickle.load(f)
     with open('encoder.pkl', 'rb') as f:
        encoder = pickle.load(f)
     with open('imputer_numeric.pkl', 'rb') as f:
        imputer_numeric = pickle.load(f)
     with open('imputer_categorical.pkl', 'rb') as f:
        imputer_categorical = pickle.load(f)

     df_input[numeric_cols_original] = imputer_numeric.transform(df_input[numeric_cols_original])
     df_input[categorical_cols_original] = imputer_categorical.transform(df_input[categorical_cols_original])
     df_input[numeric_cols_original] = scaler.transform(df_input[numeric_cols_original])

     encoded_cols_input = pd.DataFrame(encoder.transform(df_input[categorical_cols_original]))
     encoded_cols_input.columns = encoder.get_feature_names_out(categorical_cols_original)
     df_processed = pd.concat([df_input.drop(categorical_cols_original, axis=1), encoded_cols_input], axis=1)

     # переконайтеся, що порядок стовпців відповідає тренувальному набору
     expected_cols = ['вік', 'дохід', 'стать_жінка', 'стать_чоловік', 'стать_None', 'місто_Київ', 'місто_Львів', 'місто_Одеса', 'місто_Харків', 'освіта_бакалавр', 'освіта_доктор', 'освіта_магістр', 'освіта_середня']
     for col in expected_cols:
        if col not in df_processed.columns:
           df_processed[col] = 0 # заповнення відсутніх стовпців нулями

     return df_processed[expected_cols] # повертаємо стовпці у правильному порядку

  # приклад використання завантаженої моделі
  user_data = {'вік': 30, 'дохід': 55000, 'стать': 'жінка', 'місто': 'Львів', 'освіта': 'магістр'}
  processed_data = preprocess_input(user_data)
  prediction = loaded_model.predict(processed_data)
  print(f"Прогноз моделі: {prediction}")
```

Важливо: Також потрібно буде зберегти об'єкти для попередньої обробки (scaler, encoder, imputer_numeric, imputer_categorical) окремо за допомогою pickle після їхнього навчання у вашому Colab-ноутбуці та завантажити їх у застосунку для обробки вхідних даних користувача.

Налаштування для перенавчання моделі:

1. Новий набір даних у форматі, який використовувався для навчання (словник Python, що потім перетворюється у DataFrame).
2. Виконати ті самі кроки попередньої обробки даних (заповнення пропусків, масштабування, кодування).
3. Використати код для навчання моделі (LinearSVC) з оптимальними гіперпараметрами, знайденими за допомогою GridSearchCV: {'C': 1, 'dual': False, 'loss': 'squared_hinge', 'penalty': 'l1'}.
4. Зберегти перенавчену модель.

Інструкція для користувача:

1. Введіть свої дані у відповідні поля (вік, дохід, стать, місто, освіта).
2. Натисніть кнопку "Прогнозувати".
3. Модуль обробить ваші дані та покаже результат прогнозу.

Технічний звіт:

1. Опис моделі, даних та методів оптимізації:

1) Модель: Використано модель лінійного Support Vector Classifier (LinearSVC) для задачі бінарної класифікації.
2) Дані: Набір даних містив інформацію про вік, дохід, стать, місто проживання та рівень освіти користувачів. Цільовою змінною була бінарна змінна (цільова_змінна). Набір даних був невеликим, що слід враховувати при інтерпретації результатів.
3) Методи оптимізації: Для оптимізації гіперпараметрів моделі використовувався метод GridSearchCV з перехресною перевіркою (3 фолди). Це дозволило знайти найкращу комбінацію параметрів, яка максимізує F1-міру на валідаційних даних.

2. Пояснення вибору моделі та гіперпараметрів:

1) Вибір моделі: Модель LinearSVC була обрана після порівняння кількох класифікаторів (Logistic Regression, LinearSVC, Decision Tree, RandomForestClassifier) за результатами перехресної перевірки. LinearSVC показала найкращі середні значення точності та F1-міри на валідаційних фолдах.
2) Вибір гіперпараметрів: За допомогою GridSearchCV були знайдені наступні оптимальні гіперпараметри для LinearSVC: {'C': 1, 'dual': False, 'loss': 'squared_hinge', 'penalty': 'l1'}. Параметр C контролює силу регуляризації, 'l1' штраф сприяє розрідженню ознак, 'squared_hinge' є функцією втрат, а dual=False використовується для роботи з 'l1' штрафом.

3. Аналіз продуктивності та тестових результатів:

1) На тестовому наборі модель LinearSVC з оптимальними гіперпараметрами показала точність 1.00 та F1-міру 1.00.
2) Матриця плутанини на тестовому наборі показала, що обидва приклади були класифіковані правильно (один приклад класу 0 та один приклад класу 1).
3) Слід зазначити, що тестовий набір містив лише 2 приклади, тому ці результати можуть не повністю відображати продуктивність моделі на більшій кількості невідомих даних.