Київський національний університет імені Тараса Шевченка

**Лабораторна робота №3 «** **Czech Financial Dataset»**

**З навчального курсу «Розробка бізнес-аналітичних систем»**

Виконала студентка 4 курсу групи ТТП-4 Малашкіна Яна

**Вступ**

Цей звіт представляє аналіз датасету Chech Bank за допомогою чотирьох задач Data Mining. Метою було проведення прогнозування та/або класифікації (включно з кластеризацією та виявленням залежностей) залежно від специфіки даних. Для однієї задачі були застосовані різні моделі для порівняння їхньої продуктивності.

**Умова**

Knowledge Discovery  
 Develop 4 Data Mining tasks of your choice over the dataset from the Lab1: make forecasts and/or classification (clustering, finding dependencies in data or other) - depending on your data source and its specifics.  
 Try to apply (use, design) different models for the same task to compare their accuracy (performance).  
  
 The report should contain

· the tasks designed

· the models used to solve

· the obtained results (accuracy, F1, etc.)

· screenshots if necessary

· code attached (or link to Google Colab etc.)

Виявлення знань

Розробіть 4 задачі Data Mining на ваш вибір на основі набору даних з Лабораторної роботи 1: зробіть прогнози та/або класифікацію (кластеризацію, пошук залежностей у даних тощо) - залежно від вашого джерела даних та його специфіки.

Спробуйте застосувати (використати, розробити) різні моделі для однієї і тієї ж задачі, щоб порівняти їх точність (продуктивність).

Звіт повинен містити

· поставлені задачі

· моделі, використані для розв'язання

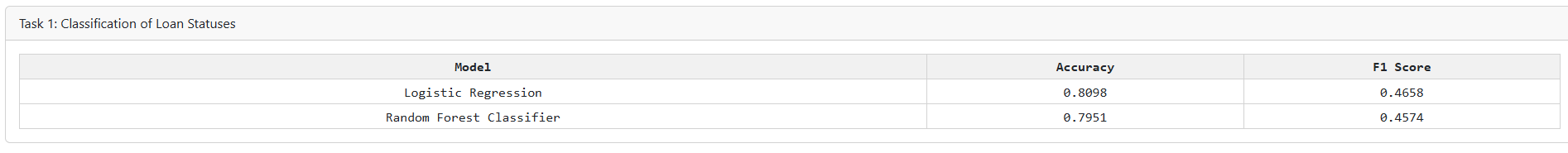
· отримані результати (точність, F1 тощо)

· скріншоти, якщо необхідно

· прикріплений код (або посилання на Google Colab тощо)

**Опис алгоритмів**

**1.1 Алгоритми класифікації (Task 1: Classification of Loan Statuses)**

****

**X = data[['amount', 'duration', 'payments', 'client\_age', 'loan\_district', 'account\_date']]**

**y = data['status']**

* Логістична регресія

Логістична регресія є одним із найпоширеніших алгоритмів для задач бінарної класифікації. Вона базується на лінійній моделі, яка використовує логістичну (сигмоїдну) функцію для оцінки ймовірності належності об'єкта до певного класу. Результат моделі — це значення в межах від 0 до 1, яке інтерпретується як ймовірність належності до позитивного класу. Цей підхід добре працює для задач, де залежність між вхідними змінними та цільовим значенням є лінійною або близькою до лінійної.

print("Навчання Logistic Regression...")

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

lr = LogisticRegression(max\_iter=2000, n\_jobs=-1)

lr.fit(X\_train\_scaled, y\_train)

y\_pred\_lr = lr.predict(X\_test\_scaled)

accuracy\_lr = accuracy\_score(y\_test, y\_pred\_lr)

f1\_lr = f1\_score(y\_test, y\_pred\_lr, average='macro')

* Класифікатор Random Forest

Random Forest (випадковий ліс) — це метод ансамблевого навчання, який поєднує велику кількість дерев рішень, щоб підвищити точність класифікації та зменшити ризик перенавчання. Кожне дерево тренується на випадковій підмножині даних і використовує випадкову підмножину ознак. Остаточний прогноз визначається голосуванням усіх дерев. Такий підхід забезпечує високу стійкість до шуму, ефективно справляється з великими обсягами даних та може моделювати складні, нелінійні залежності між ознаками.

print("Навчання Random Forest Classifier...")

rf = RandomForestClassifier(n\_estimators=100, random\_state=42, n\_jobs=-1)

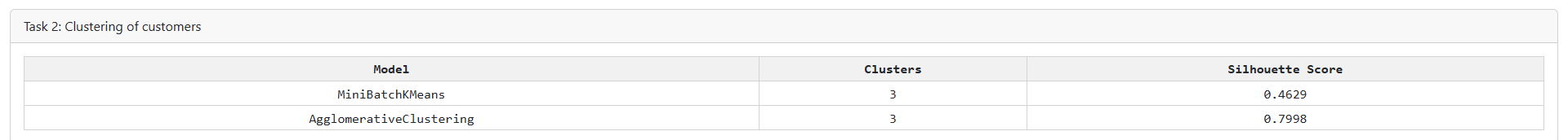
rf.fit(X\_train, y\_train)

y\_pred\_rf = rf.predict(X\_test)

accuracy\_rf = accuracy\_score(y\_test, y\_pred\_rf)

f1\_rf = f1\_score(y\_test, y\_pred\_rf, average='macro')

**1.2 Алгоритми кластеризації**



X = features[['age', 'total\_transactions', 'avg\_transaction\_amount', 'avg\_balance', 'has\_loan', 'avg\_loan\_amount', 'has\_card']]

* MiniBatchKMeans

MiniBatchKMeans — це оптимізований варіант класичного алгоритму KMeans, розроблений спеціально для роботи з великими обсягами даних. Основна ідея полягає в тому, що замість обробки всієї вибірки одразу, алгоритм працює з невеликими випадковими підмножинами даних (батчами). Це дозволяє значно зменшити час обчислень і пришвидшити збіжність, зберігаючи при цьому достатньо високу якість кластеризації. Алгоритм добре масштабується та часто використовується в задачах попередньої обробки або групування великих масивів інформації.

print("Навчання MiniBatchKMeans кластеризатора...")

kmeans = MiniBatchKMeans(n\_clusters=3, random\_state=42, batch\_size=10000)

labels\_km = kmeans.fit\_predict(X)

print("Обчислення Silhouette Score для MiniBatchKMeans...")

silhouette\_km = silhouette\_score(X, labels\_km)

* Agglomerative Clustering

Agglomerative Clustering — це метод ієрархічної кластеризації, який працює за принципом «знизу вгору». Спочатку кожна точка розглядається як окремий кластер. Далі на кожному кроці об’єднуються два найближчі кластери (тобто ті, між якими найменша відстань), і цей процес продовжується до тих пір, поки всі точки не об’єднаються в один великий кластер або не буде досягнуто заданої кількості кластерів.

print("Навчання AgglomerativeClustering кластеризатора...")

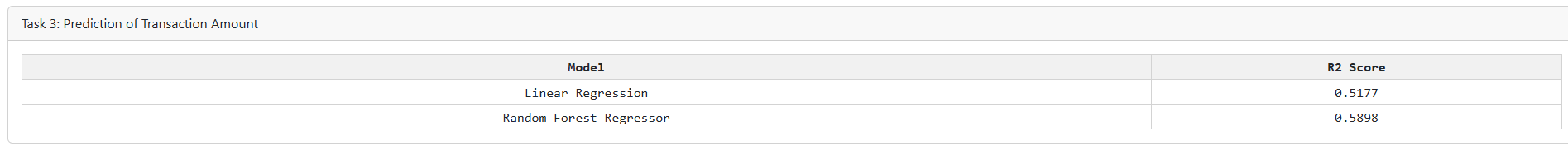
agg = AgglomerativeClustering(n\_clusters=3)

labels\_agg = agg.fit\_predict(X)

print("Обчислення Silhouette Score для AgglomerativeClustering...")

silhouette\_agg = silhouette\_score(X, labels\_agg)

**1.3 Prediction of Transaction Amount**

****

**X\_train = past[['trans\_count', 'balance', 'client\_age', 'frequency\_encoded', 'region\_encoded']]**

**Y\_train = past['trans\_amount']**

**X\_test = present[['trans\_count', 'balance', 'client\_age', 'frequency\_encoded', 'region\_encoded']]**

**Y\_test = present['trans\_amount']**

**# use the same features for real future prediction**

**X\_future = future[['trans\_count', 'balance', 'client\_age', 'frequency\_encoded', 'region\_encoded']]**

* Лінійна регресія

Лінійна регресія є базовим методом для прогнозування числових значень. Вона моделює залежність між однією або кількома незалежними змінними та цільовою змінною за допомогою прямої лінії. Основна мета — знайти такі коефіцієнти, які мінімізують різницю між передбаченими та фактичними значеннями. Цей алгоритм простий у реалізації, добре інтерпретується та ефективний у випадках, коли зв'язок між змінними є лінійним.

print("Навчання Linear Regression...")

lr = LinearRegression()

lr.fit(X\_train, Y\_train) # train the model on the past data

y\_pred\_lr = lr.predict(X\_test) # predict the test in present year

r2\_lr = r2\_score(Y\_test, y\_pred\_lr)

future['lr\_trans\_amount'] = lr.predict(X\_future) # predict the real future

* Random Forest Regressor

Random Forest Regressor — це ансамблевий метод для вирішення задач регресії, який поєднує прогнози багатьох незалежних дерев рішень. Кожне дерево створюється на різних підмножинах даних, а кінцевий прогноз є середнім значенням результатів усіх дерев. Такий підхід дозволяє зменшити ризик перенавчання та забезпечує високу точність, особливо у випадках з великою кількістю ознак або складними нелінійними залежностями.

print("Навчання Random Forest Regressor...")

rf = RandomForestRegressor(n\_estimators=100, random\_state=42, n\_jobs=-1)

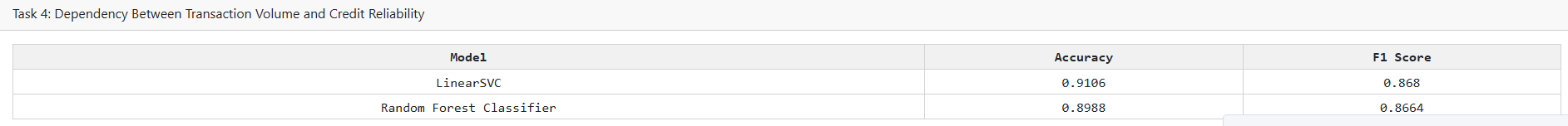
rf.fit(X\_train, Y\_train)

y\_pred\_rf = rf.predict(X\_test)

r2\_rf = r2\_score(Y\_test, y\_pred\_rf)

future['rf\_trans\_amount'] = rf.predict(X\_future)

**1.4 Dependency Between Transaction Volume and Credit Reliability**

****

**X = features[['total\_transactions', 'avg\_balance']]**

**Y = features['loan\_status']**

* Лінійна регресія (із крос-валідацією)

Для дослідження залежності між обсягом транзакцій та кредитною надійністю може використовуватись лінійна регресія з крос-валідацією (CV). Крос-валідація дозволяє оцінити стабільність та узагальнювальну здатність моделі, зменшуючи ризик перенавчання. Такий підхід дозволяє виявити, чи існує статистично значущий лінійний зв'язок між змінними.

print("Навчання LinearSVC ...")

# Use a pipeline to ensure the features are scaled (important for SVMs)

svc = make\_pipeline(StandardScaler(), LinearSVC(random\_state=42, max\_iter=10000))

svc.fit(X\_train, Y\_train) # Train the model on 70%

y\_pred\_svc = svc.predict(X\_test) # Asking the already trained model to predict the remaining 30%

accuracy\_svc = accuracy\_score(Y\_test, y\_pred\_svc)

f1\_svc = f1\_score(Y\_test, y\_pred\_svc, average='weighted')

* Random Forest Classifier

У цій задачі класифікатор Random Forest застосовується для передбачення рівня кредитної надійності на основі обсягів транзакцій та інших параметрів. Завдяки своїй здатності моделювати складні залежності та враховувати взаємодію між ознаками, алгоритм може виявляти закономірності, які важко помітити за допомогою простіших моделей. Класифікація здійснюється на основі голосування дерев, що забезпечує стабільні та надійні результати.

print("Навчання Random Forest Classifier...")

rf = RandomForestClassifier(n\_estimators=100, random\_state=42, n\_jobs=-1)

rf.fit(X\_train, Y\_train)

y\_pred\_rf = rf.predict(X\_test)

accuracy\_rf = accuracy\_score(Y\_test, y\_pred\_rf)

f1\_rf = f1\_score(Y\_test, y\_pred\_rf, average='weighted')

**Висновок**

Загальний аналіз результатів

У цьому проєкті було розглянуто різні задачі Data Mining — класифікація, кластеризація та регресія — з використанням кількох популярних алгоритмів. Для кожного завдання визначався "лідер-модель" за відповідною метрикою (Accuracy, F1 Score, R2, Silhouette Score), проте загалом різниця між результатами моделей була не критичною, що свідчить про збалансованість даних та правильний вибір ознак.

Варто зазначити, що якість моделі напряму залежить від якості вхідних даних. У кожному методі важливо правильно підібрати та підготувати ознаки, оскільки навіть найпотужніший алгоритм не дасть хорошого результату без релевантної інформації.

У класифікації (Task 1 і Task 4) обидва методи дали подібні значення F1 Score, але LinearSVC дещо випередив Random Forest.

У кластеризації (Task 2) AgglomerativeClustering мав вищий Silhouette Score, що вказує на чіткішу кластерну структуру.

У регресійній задачі (Task 3) Random Forest Regressor показав кращий R² результат, що свідчить про кращу здатність моделювати залежності.

Отже, ключовим фактором ефективного аналізу є не тільки вибір моделі, а й правильна підготовка та розуміння даних. Комбінація алгоритмів і ретельна обробка дозволяють досягти надійних результатів у фінансовому аналізі транзакцій.