

BIG DATA ANALYTICS

Enseigné par : Mr. Hmida HMIDA

Travail réalisée par :

Sondes AHMED

Israa ETTISS



FinTRAFIC

FinTRAFIC

Une approche intégrée pour améliorer la performance, la ponctualité et la planification

Contexte

Importance du transport public pour la mobilité urbaine et régionale en Finlande.



Défis actuels :

Retards, accidents, mauvaise synchronisation entre différents modes de transport.

Objectifs du Projet :



Un projet illustrant une Application BigDATA pour le transpor public en Finlands



Collecte de données

Centraliser les données pour une analyse complète.



Analyse descriptive

Comprendre les performances actuelles du réseau de transport.



Analyse prédictive

Anticiper les retards en temps réel.



Analyse de graphe

Identifier les lacunes dans la planification et la synchronisation des services.

Vue d'Ensemble de l'Architecture:



Sources de Données: API Digitransit (<https://digitransit.fi/en/developers/apis/>)



Ingestion des Données

Apache Kafka: Plateforme de streaming pour la collecte et la transmission en temps réel des données.



Stockage des Données

MongoDB: Base de données NoSQL pour le stockage flexible et évolutif des données structurées et non structurées.



Traitement des Données:

Apache Spark: Framework de traitement big data pour les analyses descriptives, et prédictives.

Vue d'Ensemble de l'Architecture:



Sources de Données: API Digitransit (<https://digitransit.fi/en/developers/apis/>)



Analyse en Temps Réel:

Spark Streaming: Pour les prédictions en temps réel des retards.



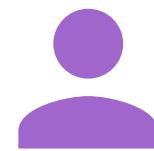
Power BI

Visualisation et Reporting

Power BI pour la création de dashboards interactifs



Solutions



Pipeline de Stockage des Données et visualisation



Ingestion des Données:

API de Digitransi +
Kafka



Transmission et Buffering

Kafka



Stockage:

MongoDB



Visualisation des Données:

PowerBI



kafka



 mongoDB



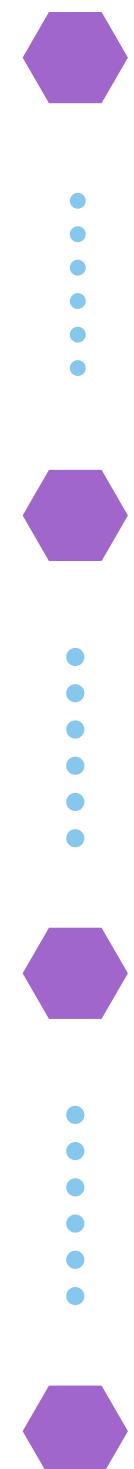
 Power BI



Solutions

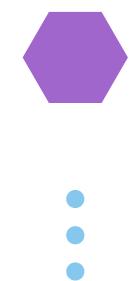


Pipeline de Prédiction en Streaming



Ingestion des Données:

API de Digitransi +
Kafka



Traitements en Streaming

Spark

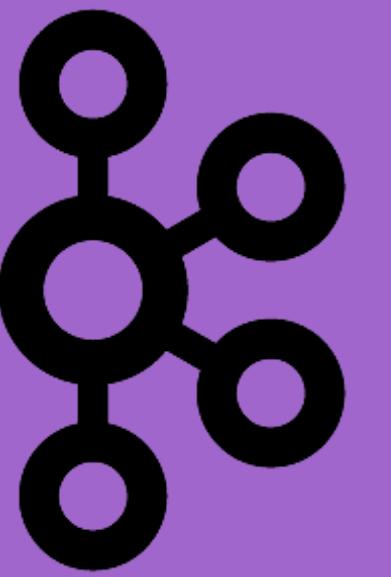


Modélisation Prédictive

Modèles de deep
Learning



TensorFlow



kafka

jupyter producer Last Checkpoint: last month

File Edit View Run Kernel Settings Help



Trusted

Code ▾

JupyterLab ▾ Python 3 (ipykernel) ⚡

```
# Streaming loop: Fetch data and send it to Kafka every 30 seconds
while True:
    fetch_and_send()
    time.sleep(30)
```

```
Fetched data from API: {
  "data": {
    "plan": {
      "itineraries": [
        {
          "walkDistance": 308.68238462365167,
          "duration": 850,
          "legs": [
            {
              "mode": "WALK",
              "startTime": 1735812007000,
              "endTime": 1735812050000,
              "from": {
                "lat": 60.19775,
                "lon": 24.94053,
                "name": "Origin",
                "stop": null
              },
              "to": {
                "lat": 60.1884,
                "lon": 25.00744,
                "name": "Destination",
                "stop": null
              }
            }
          ]
        }
      ]
    }
  }
}
```

[]:

jupyter consumer Last Checkpoint: last month

File Edit View Run Kernel Settings Help

Trusted

Code ▾

JupyterLab ▾ Python 3 (ipykernel) ⚡

```
else:
    print("Empty message received")

except KeyboardInterrupt:
    print("Consumer interrupted")
finally:
    consumer.close()

# Start consuming messages
consume_messages()

cheduledDeparture": 44040}]]]}, "to": {"lat": 60.1884, "lon": 25.00744, "name": "Destination", "stop": null}, "trip": null}]]}]}
Processed data: {
  "data": {
    "plan": {
      "itineraries": [
        {
          "walkDistance": 308.68238462365167,
          "duration": 850,
          "legs": [
            {
              "mode": "WALK",
              "startTime": 1735812007000,
              "endTime": 1735812050000,
              "from": {
                "lat": 60.19775,
                "lon": 24.94053,
                "name": "Origin",
                "stop": null
              },
              "to": {
                "lat": 60.1884,
                "lon": 25.00744,
                "name": "Destination",
                "stop": null
              }
            }
          ]
        }
      ]
    }
  }
}
```

[]:

jupyter producer Last Checkpoint: last month

File Edit View Run Kernel Settings Help



Trusted

Code ▾

JupyterLab ▾ Python 3 (ipykernel) ⚡

```
# Streaming loop: Fetch data and send it to Kafka every 30 seconds
while True:
    fetch_and_send()
    time.sleep(30)
```

```
Fetched data from API: {
  "data": {
    "plan": {
      "itineraries": [
        {
          "walkDistance": 308.68238462365167,
          "duration": 850,
          "legs": [
            {
              "mode": "WALK",
              "startTime": 1735812007000,
              "endTime": 1735812050000,
              "from": {
                "lat": 60.19775,
                "lon": 24.94053,
                "name": "Origin",
                "stop": null
              },
              "to": {
                "lat": 60.1884,
                "lon": 25.00744,
                "name": "Destination",
                "stop": null
              }
            }
          ]
        }
      ]
    }
  }
}
```

[]:

jupyter consumer Last Checkpoint: last month

File Edit View Run Kernel Settings Help

Trusted

Code ▾

JupyterLab ▾ Python 3 (ipykernel) ⚡

```
else:
    print("Empty message received")

except KeyboardInterrupt:
    print("Consumer interrupted")
finally:
    consumer.close()

# Start consuming messages
consume_messages()

cheduledDeparture": 44040}]]]}, "to": {"lat": 60.1884, "lon": 25.00744, "name": "Destination", "stop": null}, "trip": null}]]}]}
Processed data: {
  "data": {
    "plan": {
      "itineraries": [
        {
          "walkDistance": 308.68238462365167,
          "duration": 850,
          "legs": [
            {
              "mode": "WALK",
              "startTime": 1735812007000,
              "endTime": 1735812050000,
              "from": {
                "lat": 60.19775,
                "lon": 24.94053,
                "name": "Origin",
                "stop": null
              },
              "to": {
                "lat": 60.1884,
                "lon": 25.00744,
                "name": "Destination",
                "stop": null
              }
            }
          ]
        }
      ]
    }
  }
}
```

[]:

Spark 

ETL avec SparkStream

```
10/adjust logging level use sc.setLogLevel(newLevel). For SparkUI, use setLogLevel(newLevel).
25/01/02 09:39:49 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
Spark Session created successfully

[2]: df_streaming = spark.readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:9092") \
    .option("subscribe", "digitransit_data") \
    .load()

df_streaming = df_streaming.selectExpr("CAST(value AS STRING)")
df_streaming.printSchema()

root
 |-- value: string (nullable = true)
```

```
[3]: from pyspark.sql import functions as F
```

```
[4]: from pyspark.sql.functions import col, coalesce, struct, transform, lit, expr, to_json
df_transformed = df_parsed.selectExpr("""
    STRUCT(
        STRUCT(
            TRANSFORM(data.data.plan.itineraries,
                itinerary -> STRUCT(
                    COALESCE(itinerary.walkDistance, 0.0) AS walkDistance,
                    COALESCE(itinerary.duration, 0) AS duration,
                    TRANSFORM(itinerary.legs,
                        leg -> STRUCT(
                            COALESCE(leg.mode, '') AS mode,
                            COALESCE(CAST(leg.startTime / 1000 AS timestamp), '1970-01-01 00:00:00') AS startTime,
                            COALESCE(CAST(leg.endTime / 1000 AS timestamp), '1970-01-01 00:00:00') AS endTime,
                            COALESCE(leg.from.lat, 0.0) AS from_lat,
                            COALESCE(leg.from.lon, 0.0) AS from_lon,
                            COALESCE(leg.from.name, '') AS from_name,
                            TRANSFORM(leg.from.stop.patterns,
                                pattern -> STRUCT(COALESCE(pattern.code, '') AS code)
                            ) AS from_pattern,
                            COALESCE(leg.to.lat, 0.0) AS to_lat,
                            COALESCE(leg.to.lon, 0.0) AS to_lon,
                            COALESCE(leg.to.name, '') AS to_name,
                            TRANSFORM(leg.to.stop.patterns,
                                pattern -> STRUCT(COALESCE(pattern.code, '') AS code)
                            ) AS to_pattern,
                            COALESCE(leg.trip.gtfsId, '') AS gtfsId,
                            COALESCE(leg.trip.pattern.trip_pattern_code, '') AS trip_pattern_code,
                            COALESCE(leg.trip.tripHeadsign, '') AS tripHeadsign
                        )
                    ) AS legs
                ) AS itineraries
            ) AS plan
        ) AS data
    """
)
```

jupyter ETL Last Checkpoint: 23 days ago

File Edit View Run Kernel Settings Help

JupyterLab ▾ Python 3 (ipykernel) Trusted

```
[*]: def process_batch(df, batch_id):
    df.write \
        .format("mongo") \
        .option("spark.mongodb.output.uri", "mongodb+srv://sondesahmed77:93ZVakay5j4UVyJ5@cluster0.hd18x.mongodb.net/DigitTransitAPI.clean_dat")
    .mode("append") \
    .save()

    # Write the streaming data and apply the batch processing function
    df_transformed.writeStream \
        .foreachBatch(process_batch) \
        .outputMode("append") \
        .start() \
        .awaitTermination()

25/01/02 06:37:16 WARN ResolveWriteToStream: Temporary checkpoint location created which is deleted normally when the query didn't fail: /tmp/p/temporary-f8b3cfaa-5622-4286-be42-2ef2c10906c8. If it's required to delete it under any circumstances, please set spark.sql.streaming.forceDeleteTempCheckpointLocation to true. Important to know deleting temp checkpoint folder is best effort.
25/01/02 06:37:16 WARN ResolveWriteToStream: spark.sql.adaptive.enabled is not supported in streaming DataFrames/Datasets and will be disabled.
25/01/02 06:37:18 WARN SparkStringUtils: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.sql.debug.maxToStringFields'.
```



Stockage de données dans MongoDB

The screenshot shows the MongoDB Atlas Data Services interface. The left sidebar navigation includes: Overview, DATABASE (selected), Clusters, SERVICES, SECURITY, and Advanced. The main area displays the **DigitTransitAPI** cluster's storage statistics: STORAGE SIZE: 440KB, LOGICAL DATA SIZE: 2.88MB, TOTAL DOCUMENTS: 769, INDEXES TOTAL SIZE: 52KB. It features tabs for Find, Indexes, Schema Anti-Patterns (0), Aggregation, and Search Indexes. A search bar at the top right says "Search Namespaces". Below the stats, there's a section to "Generate queries from natural language in Compass" and an "INSERT DOCUMENT" button. The "Find" tab is active, showing a query builder with "Type a query: { field: 'value' }" and a "Reset" and "Apply" button. The results pane shows a single document:

```
0: Object
  mode : "WALK"
  startTime : "2024-12-12 13:20:02"
  endTime : "2024-12-12 13:20:45"
  from_lat : 60.19775
  from_lon : 24.94053
  from_name : "Origin"
  to_lat : 60.198034
  to_lon : 24.940222
  to_name : "Asemapäälikönkatu"
  to_pattern : Array (2)
    gtfsId : ""
    trip_pattern_code : ""
    trip_headsign : ""
```

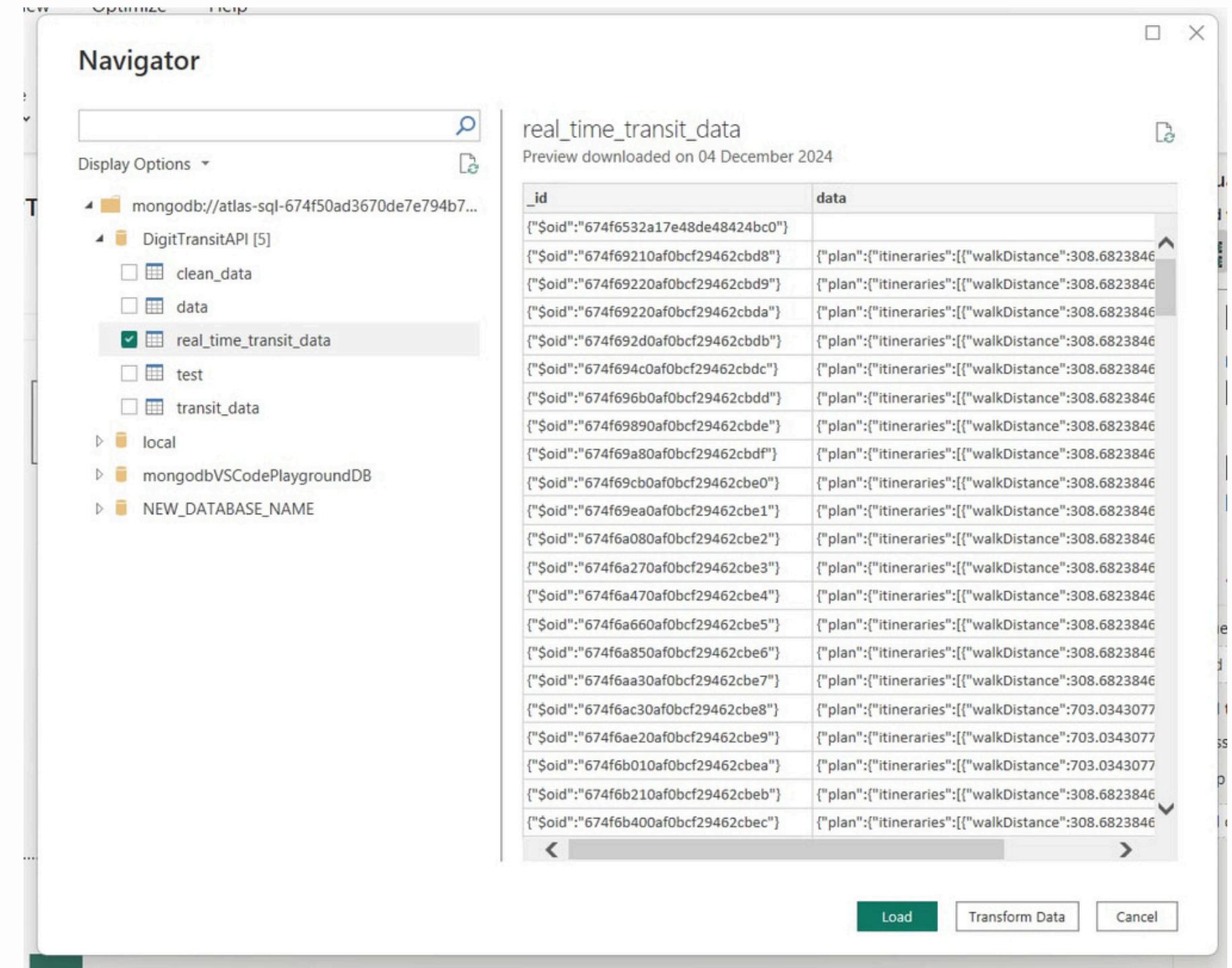
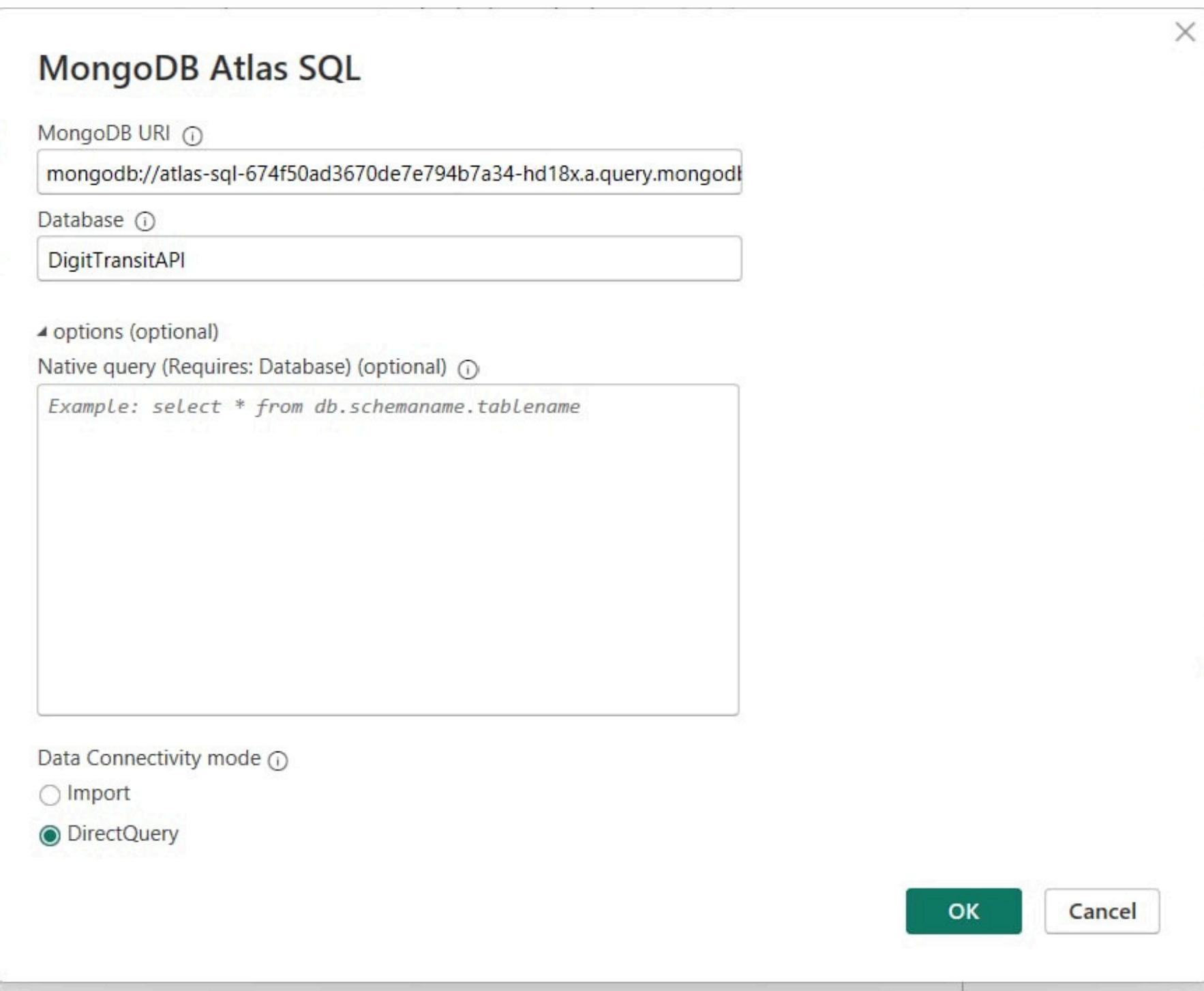
Navigation buttons include PREVIOUS, 1-20 of many results, and NEXT.



Power BI

Connexion Du MongoDB et PowerBI

Récupération des données



Transformations avec l'outil Query Editor de PowerBI

The screenshot shows the PowerBI Query Editor interface with the following details:

- Queries [3]**: A sidebar on the left listing three queries: **Iteraty** (selected), **Patterns**, and **Legs**.
- Query Editor Area**: The main area displays a table with three columns:
 - A**
 - _id**
 - Walk Distance**
 - Duration**The table contains 21 rows of data.
- Query Settings**: A panel on the right containing:
 - PROPERTIES**: Name is set to **Iteraty**.
 - APPLIED STEPS**: A list of transformations applied to the query, with **Removed Duplicates1** highlighted.

	A	_id	Walk Distance	Duration
1		675b273de2b139501b6e...	308.6823846	830
2		675b273de2b139501b6e...	703.0343077	1380
3		675b273de2b139501b6e...	308.6823846	786
4		675b2754e2b139501b6e...	308.6823846	830
5		675b2754e2b139501b6e...	703.0343077	1380
6		675b2754e2b139501b6e...	308.6823846	786
7		675b2774e2b139501b6e...	308.6823846	830
8		675b2774e2b139501b6e...	703.0343077	1353
9		675b2774e2b139501b6e...	308.6823846	786
10		675b2794e2b139501b6e...	308.6823846	830
11		675b2794e2b139501b6e...	703.0343077	1353
12		675b2794e2b139501b6e...	308.6823846	786
13		675b27b3e2b139501b6e...	308.6823846	830
14		675b27b3e2b139501b6e...	703.0343077	1353
15		675b27b3e2b139501b6e...	308.6823846	786
16		675b27d2e2b139501b6e...	308.6823846	850
17		675b27d2e2b139501b6e...	703.0343077	1353
18		675b27d2e2b139501b6e...	308.6823846	799
19		675b27f0e2b139501b6e...	308.6823846	850
20		675b27f0e2b139501b6e...	703.0343077	1353
21		675b27f0e2b139501b6e...	308.6823846	799

Queries [3]

= Table.Distinct(#"Replaced Value")

	A ^B _C _id	ABC 123 Stop Pattern	ABC 123 gtfsId	ABC 123 tripHeadsign
1	675b273de2b139501b6e...	HSL:4518:0:01		
2	675b273de2b139501b6e...	HSL:1059:1:01		
3	675b273de2b139501b6e...	HSL:1097N:0:01	HSL:1059_20241211_To_...	Herttoniemi (M)
4	675b273de2b139501b6e...	HSL:1085N:0:01	HSL:1059_20241211_To_...	Herttoniemi (M)
5	675b273de2b139501b6e...	HSL:1092N:0:01	HSL:1059_20241211_To_...	Herttoniemi (M)
6	675b273de2b139501b6e...	HSL:1096N:0:01	HSL:1059_20241211_To_...	Herttoniemi (M)
7	675b273de2b139501b6e...	HSL:1090A:0:01	HSL:1059_20241211_To_...	Herttoniemi (M)
8	675b273de2b139501b6e...	HSL:1500:1:01	HSL:1059_20241211_To_...	Herttoniemi (M)
9	675b273de2b139501b6e...	HSL:1090N:0:01	HSL:1059_20241211_To_...	Herttoniemi (M)
10	675b273de2b139501b6e...	HSL:1094N:0:01	HSL:1059_20241211_To_...	Herttoniemi (M)
11	675b273de2b139501b6e...	HSL:1095N:0:01	HSL:1059_20241211_To_...	Herttoniemi (M)
12	675b273de2b139501b6e...	HSL:1087N:0:01	HSL:1059_20241211_To_...	Herttoniemi (M)
13	675b273de2b139501b6e...	HSL:1059:1:01	HSL:1059_20241211_To_...	Herttoniemi (M)
14	675b273de2b139501b6e...	HSL:9841N:0:01	HSL:1059_20241211_To_...	Herttoniemi (M)
15	675b273de2b139501b6e...	HSL:2510:1:01	HSL:1059_20241211_To_...	Herttoniemi (M)
16	675b273de2b139501b6e...			
17	675b273de2b139501b6e...	HSL:1500:1:01		
18	675b273de2b139501b6e...	HSL:2510:1:01		
19	675b273de2b139501b6e...	HSL:1097N:0:01	HSL:1500_20241211_To_...	Itäkeskus (M)
20	675b273de2b139501b6e...	HSL:1085N:0:01	HSL:1500_20241211_To_...	Itäkeskus (M)
21	675b273de2b139501b6e...	HSL:1092N:0:01	HSL:1500_20241211_To_...	Itäkeskus (M)
22	675b273de2b139501b6e...	HSL:1096N:0:01	HSL:1500_20241211_To_...	Itäkeskus (M)
23	675b273de2b139501b6e...	HSL:1090A:0:01	HSL:1500_20241211_To_...	Itäkeskus (M)
24	675b273de2b139501b6e...	HSL:1500:1:01	HSL:1500_20241211_To_...	Itäkeskus (M)
25	675b273de2b139501b6e...	HSL:1090N:0:01	HSL:1500_20241211_To_...	Itäkeskus (M)
26	675b273de2b139501b6e...	HSL:1094N:0:01	HSL:1500_20241211_To_...	Itäkeskus (M)

Query Settings

PROPERTIES

Name: Patterns

APPLIED STEPS

- Source
- Converted to Table
- Expanded Column1
- Expanded data
- Expanded data.plan
- Expanded data.plan.itineraries
- Expanded data.plan.itineraries1
- Changed Type
- Removed Columns
- Expanded data.plan.itineraries...
- Expanded data.plan.itineraries...
- Expanded data.plan.itineraries...
- Removed Columns1
- Renamed Columns
- Removed Duplicates
- Replaced Value
- Removed Duplicates1

Queries [3]

= Table.Distinct(#"Removed Columns3")

	A ^B _C _id	ABC 123 Mode	Start Time	End Time	A ^B _C Location_type	1.2 lat	1.2 lon
1	675b273de2b139501b6e...	WALK	12/12/2024 13:20:02	12/12/2024 13:20:45	From	60.19775	
2	675b273de2b139501b6e...	WALK	12/12/2024 13:20:02	12/12/2024 13:20:45	To	60.198034	
3	675b273de2b139501b6e...	BUS	12/12/2024 13:20:45	12/12/2024 13:29:55	From	60.198034	
4	675b273de2b139501b6e...	BUS	12/12/2024 13:20:45	12/12/2024 13:29:55	To	60.18864	
5	675b273de2b139501b6e...	WALK	12/12/2024 13:29:55	12/12/2024 13:33:52	From	60.18864	
6	675b273de2b139501b6e...	WALK	12/12/2024 13:29:55	12/12/2024 13:33:52	To	60.1884	
7	675b273de2b139501b6e...	WALK	12/12/2024 13:14:14	12/12/2024 13:21:35	From	60.19775	
8	675b273de2b139501b6e...	WALK	12/12/2024 13:14:14	12/12/2024 13:21:35	To	60.197939	
9	675b273de2b139501b6e...	BUS	12/12/2024 13:21:35	12/12/2024 13:33:17	From	60.197939	
10	675b273de2b139501b6e...	BUS	12/12/2024 13:21:35	12/12/2024 13:33:17	To	60.18864	
11	675b273de2b139501b6e...	WALK	12/12/2024 13:33:17	12/12/2024 13:37:14	To	60.1884	
12	675b273de2b139501b6e...	WALK	12/12/2024 13:33:17	12/12/2024 13:37:14	From	60.18864	
13	675b273de2b139501b6e...	WALK	12/12/2024 13:29:59	12/12/2024 13:30:42	To	60.198034	
14	675b273de2b139501b6e...	WALK	12/12/2024 13:29:59	12/12/2024 13:30:42	From	60.19775	
15	675b273de2b139501b6e...	BUS	12/12/2024 13:30:42	12/12/2024 13:39:08	To	60.18864	
16	675b273de2b139501b6e...	BUS	12/12/2024 13:30:42	12/12/2024 13:39:08	From	60.198034	
17	675b273de2b139501b6e...	WALK	12/12/2024 13:39:08	12/12/2024 13:43:05	From	60.18864	
18	675b273de2b139501b6e...	WALK	12/12/2024 13:39:08	12/12/2024 13:43:05	To	60.1884	
19	675b2754e2b139501b6e...	WALK	12/12/2024 13:20:02	12/12/2024 13:20:45	To	60.198034	
20	675b2754e2b139501b6e...	WALK	12/12/2024 13:20:02	12/12/2024 13:20:45	From	60.19775	
21	675b2754e2b139501b6e...	BUS	12/12/2024 13:20:45	12/12/2024 13:29:55	From	60.198034	
22	675b2754e2b139501b6e...	BUS	12/12/2024 13:20:45	12/12/2024 13:29:55	To	60.18864	
23	675b2754e2b139501b6e...	WALK	12/12/2024 13:29:55	12/12/2024 13:33:52	To	60.1884	
24	675b2754e2b139501b6e...	WALK	12/12/2024 13:29:55	12/12/2024 13:33:52	From	60.18864	
25	675b2754e2b139501b6e...	WALK	12/12/2024 13:14:14	12/12/2024 13:21:35	From	60.19775	
26	675b2754e2b139501b6e...	WALK	12/12/2024 13:14:14	12/12/2024 13:21:35	To	60.197939	
27	675b2754e2b139501b6e...	BUS	12/12/2024 13:21:35	12/12/2024 13:33:17	To	60.18864	
28							

8 COLUMNS, 999+ ROWS Column profiling based on top 1000 rows

PREVIEW DOWNLOADED AT 12:46

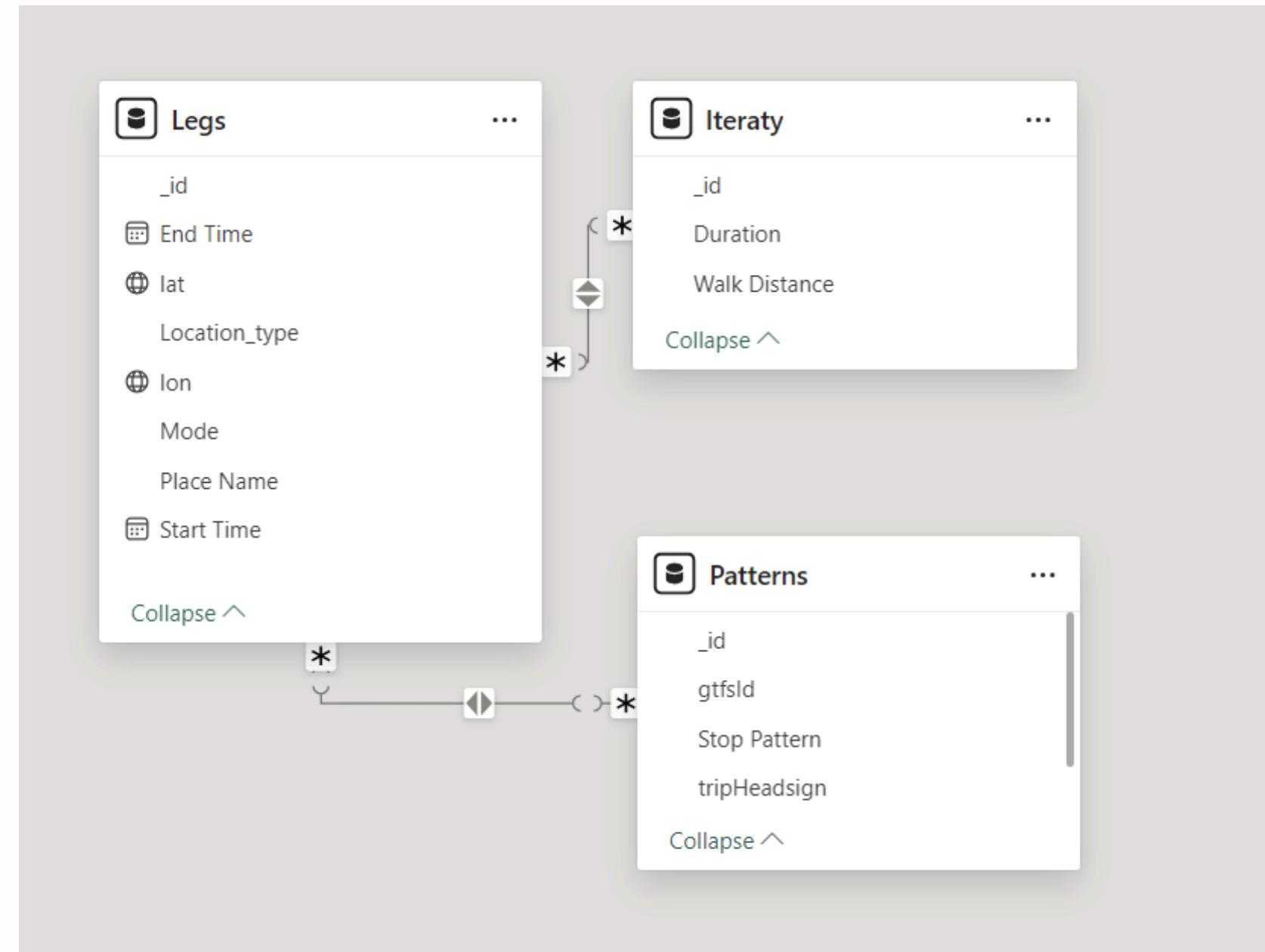
Query Settings

PROPERTIES

Name: Legs

APPLIED STEPS

- Expanded data.plan.itinera...
- Changed Type
- Removed Columns
- Expanded data.plan.itinera...
- Expanded data.plan.itinera...
- Removed Columns1
- Renamed Columns
- Changed Type1
- Removed Duplicates
- Added Index
- Unpivoted Columns
- Split Column by Delimiter
- Changed Type2
- Renamed Columns1
- Pivoted Column
- Sorted Rows
- Added Custom
- Sorted Rows1
- Removed Columns2
- Filtered Rows
- Removed Duplicates1
- Removed Columns3
- Removed Duplicates2



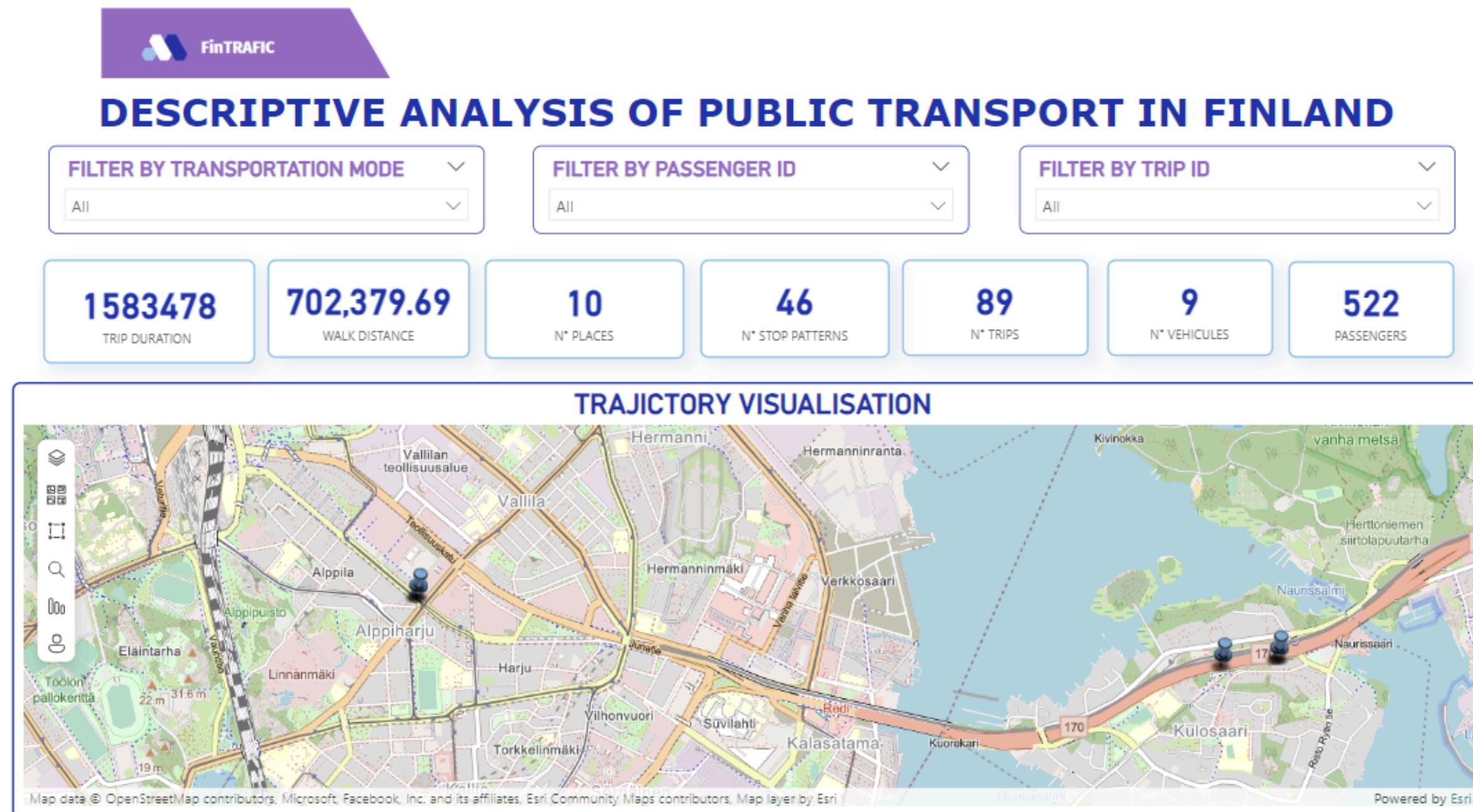
Analyse Descriptive

KPIs Principaux :

- **Vue d'ensemble des trajectoires**
- **Durée moyenne des trajets par mode**
- **Fréquence des trajets par schéma d'arrêts**

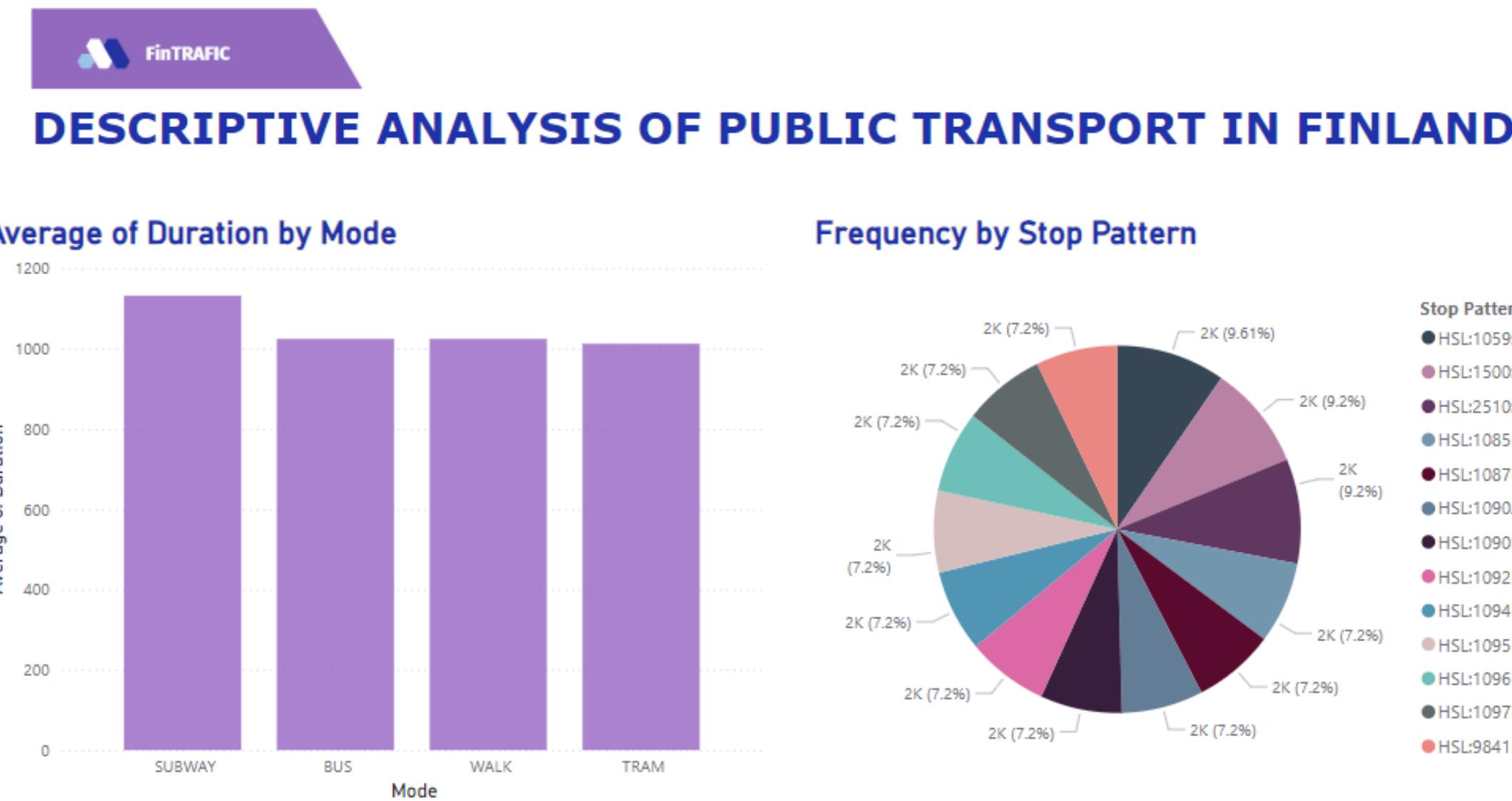
Visualisations avec PowerBI

Visualisation des trajectoires avec filtres interactifs sur le mode de transport, l'identifiant du passager et l'identifiant du voyage.



Visualisations

Moyenne des durées des trajets, regroupée par mode de transport et Fréquence des trajets, segmentée par les arrêts



Analyse Prédictive

Objectif

Anticiper les retards pour améliorer la réactivité et la planification.

Méthodologie:



Prétraitement des
Données

Sélection des
Caractéristiques

Entraînement des
Modèles

Évaluation des
Modèles

Prétraitement des Données

Étape	Description	Objectif
Chargement des données	Lecture des données JSON avec un schéma prédéfini (StructType).	Assurer une structure typée pour un traitement cohérent.
Filtrage des colonnes	Sélection des colonnes importantes (_id, source, timestamp, itineraries).	Réduire le volume de données en conservant uniquement les informations pertinentes.
Flattening des données	Utilisation <code>de explode</code> pour aplatisir les listes imbriquées (itineraries et legs).	Convertir les données hiérarchiques en format tabulaire pour les rendre exploitables.
Conversion des timestamps	Transformation des champs startTime et endTime en format lisible avec <code>from_unixtime</code> .	Faciliter l'interprétation des données temporelles.
Calcul de nouvelles colonnes	Ajout de leg_duration pour calculer la durée de chaque segment (endTime - startTime).	Générer une nouvelle caractéristique utile pour l'entraînement du modèle.
Traitement des valeurs manquantes	Remplacement des valeurs nulles dans les colonnes catégoriques par "unknown" et numériques par 0.0.	Assurer la robustesse des étapes suivantes en éliminant les valeurs manquantes.

Sélection des Caractéristiques

Étape	Description	Objectif
Encodage des colonnes catégoriques	Utilisation de StringIndexer pour convertir les colonnes catégoriques (source, mode, etc.) en indices numériques.	Préparer les données pour les algorithmes qui nécessitent des entrées numériques.
Assemblage des caractéristiques	Utilisation de VectorAssembler pour regrouper les colonnes (source_indexed, mode_indexed, etc.) en un vecteur unique.	Créer une représentation vectorielle compacte des données pour l'entraînement.
Cible de prédiction	La colonne duration est utilisée comme cible (target_duration).	Définir la variable à prédire pour l'entraînement du modèle.
Normalisation implicite	Conversion des données catégoriques et numériques en un espace homogène (vecteurs).	Assurer une homogénéité des entrées pour éviter des biais liés à l'échelle des données.
Division des données	Division des données en ensembles d'entraînement (80%) et de test (20%).	Évaluer la généralisation du modèle en utilisant un ensemble de données non vu lors de l'entraînement.

Entraînement des Modèles

Algorithme	Avantages	Inconvénients
Régression Linéaire	<ul style="list-style-type: none">- Simple à implémenter et rapide à exécuter.- Interprétable, idéal pour identifier les relations linéaires entre les variables.	<ul style="list-style-type: none">- Sensible aux outliers.- Performances limitées avec des données complexes ou des interactions non linéaires.
Random Forest Regressor	<ul style="list-style-type: none">- Gère bien les relations non linéaires et les interactions complexes.- Résistant aux outliers.- Réduit le risque de surapprentissage grâce à l'agrégation.	<ul style="list-style-type: none">- Plus lent pour les grandes bases de données.- Difficulté à interpréter le modèle (boîte noire).
GPT Regressor	<ul style="list-style-type: none">- Bonne généralisation sur des tâches complexes grâce à des capacités de traitement avancé.- Peut s'adapter à une variété de formats de données.	<ul style="list-style-type: none">- Nécessite des ressources matérielles importantes.- Difficulté à ajuster les hyperparamètres pour des tâches spécifiques.
Deep Learning	<ul style="list-style-type: none">- Capable de modéliser des relations complexes et non linéaires.- Gère de grandes quantités de données.- Peut produire des prédictions précises après optimisation.	<ul style="list-style-type: none">- Longtemps d'entraînement et besoin de puissantes ressources de calcul.- Sensible à la qualité et à la quantité des données.

Lineair Regression

```
Root Mean Squared Error (RMSE): 79.26075431582646
+-----+-----+
|      prediction|target_duration|
+-----+-----+
| 799.7724026071422|          777|
| 799.7724026071422|          777|
| 799.7724026071422|          804|
| 799.7724026071422|          804|
| 799.7724026071422|          880|
| 799.7724026071422|          834|
| 799.7724026071422|          837|
| 799.7724026071422|          837|
| 799.7724026071422|          802|
| 799.7724026071422|          768|
+-----+-----+
only showing top 10 rows
```

DNN implémenté à l'aide de TensorFlow et Keras

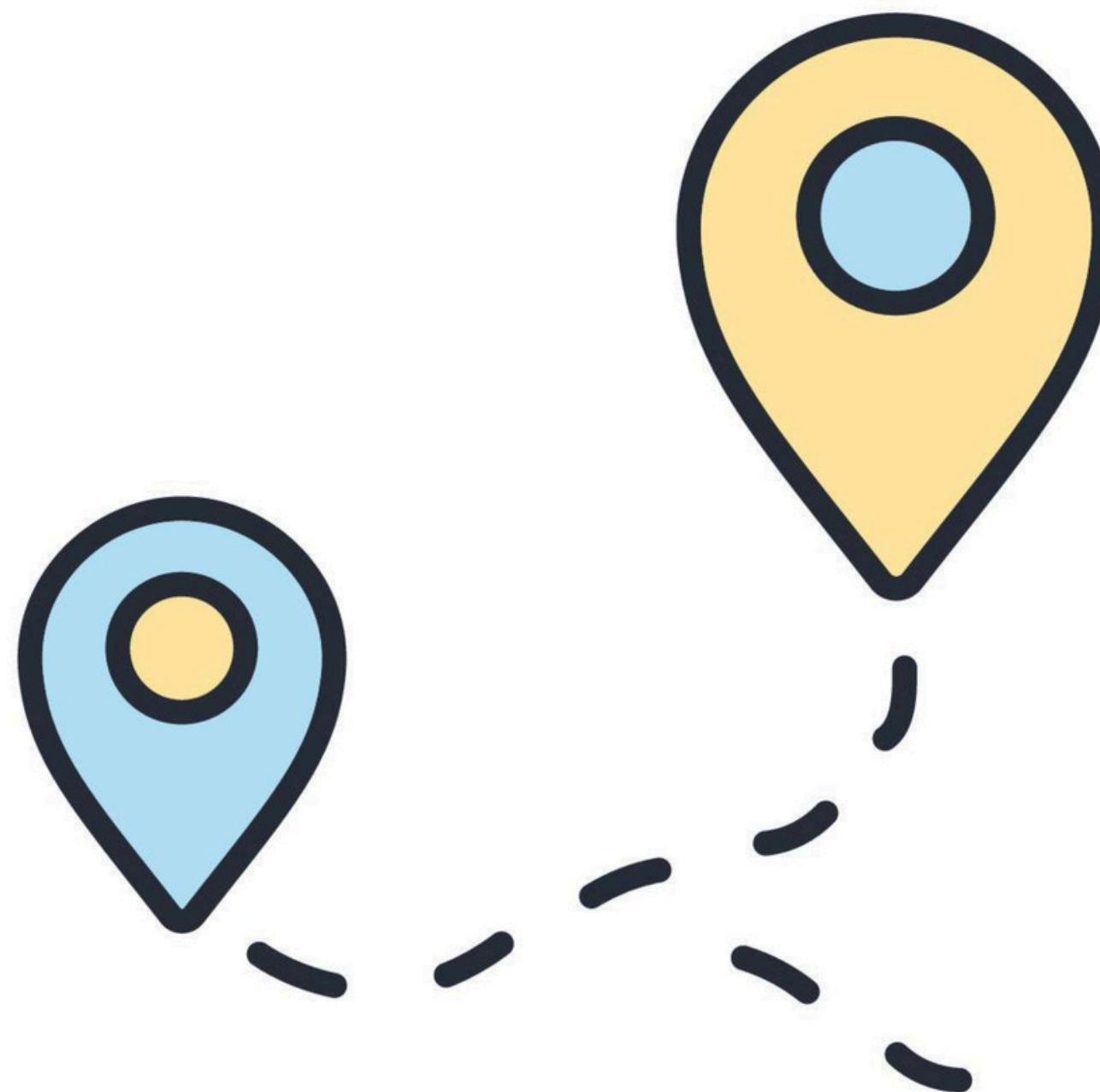
```
Prediction: 1261.68, Actual: 1259.00
Prediction: 792.28, Actual: 834.00
Prediction: 795.22, Actual: 714.00
Prediction: 1261.68, Actual: 1286.00
Prediction: 795.22, Actual: 742.00
Prediction: 1257.48, Actual: 1218.00
Prediction: 1261.68, Actual: 1259.00
Prediction: 792.28, Actual: 773.00
Prediction: 792.28, Actual: 801.00
Prediction: 1269.72, Actual: 1280.00
```

```
rmse = np.sqrt(np.mean((predictions - y_test_original) ** 2))
print(f"RMSE: {rmse:.2f}")
```

RMSE: 49.32

Évaluation des Modèles

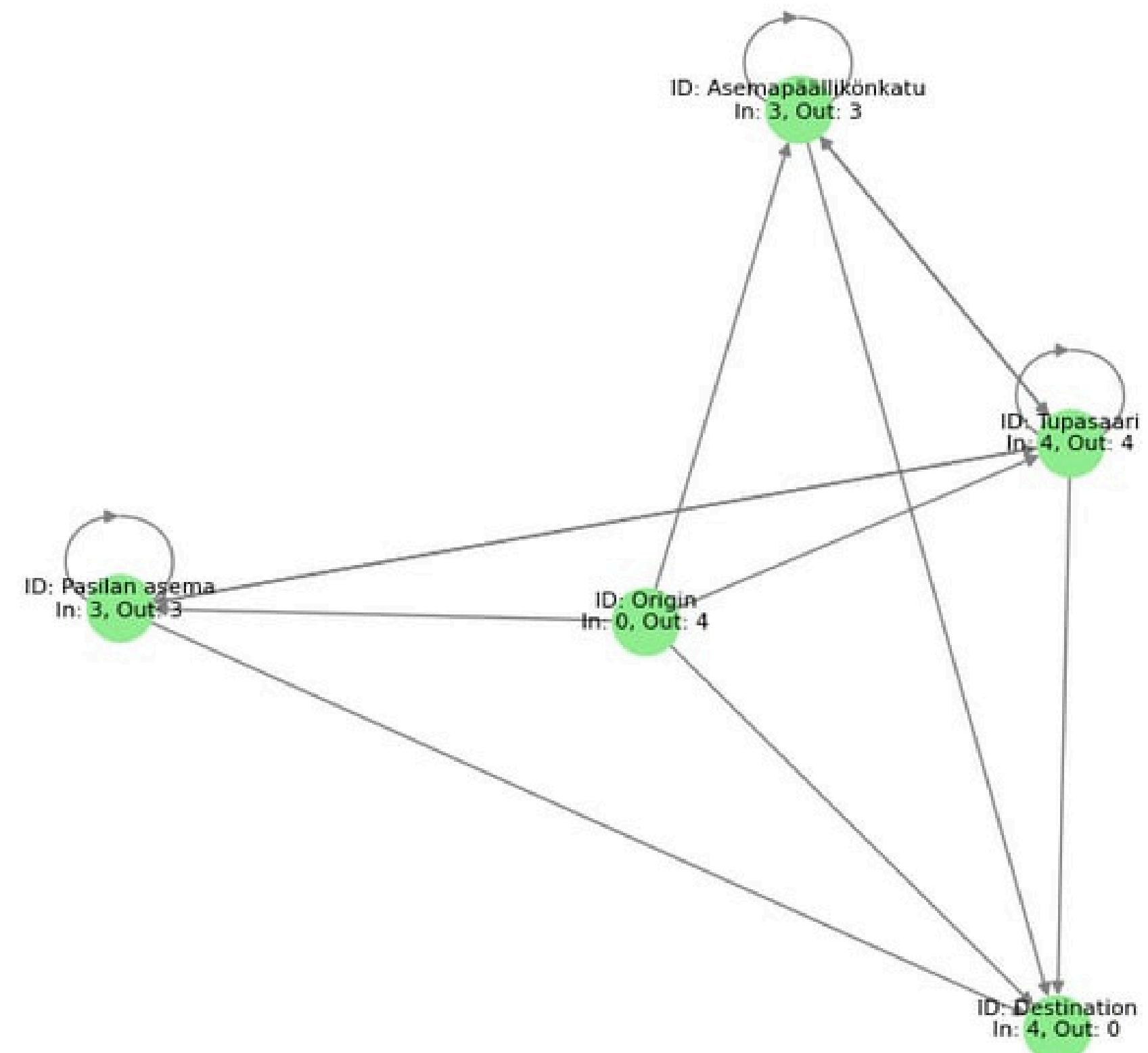
Modèle	RMSE	Comportement des Prédictions	Approches et Optimisations
Régression Linéaire	79,20	Prédiction constante (<u>799.772</u>)	Modèle simple sans optimisation.
Random Forest Regressor	70,52	Prédiction constante (<u>791.98</u>)	Utilisation d'un modèle d'ensemble basé sur des arbres de décision.
GPT Regressor	50.70	Prédiction constante (<u>772.10</u>)	Modèle basé sur GPT, offrant une meilleure performance que les précédents, mais la prédiction reste constante.
Deep Learning (Initial)	451.05	Prédictions variables	Modèle complexe basé sur le Deep Learning, mais avec un RMSE élevé.
Deep Learning (Amélioré)	337,92	Prédictions variables, mais plus proches de la réalité	Optimisations appliquées : Adam, MinMaxScaler, Dropouts, Danses. RMSE amélioré par rapport à la version initiale.
Deep Learning (Optimisé)	49,32	Prédictions variables et proches des valeurs réelles	Optimisation avancée avec Early Stopping, Sequences Shield, MinMaxScaler, Dropouts, Adam. Résultat final : RMSE de 49 et prédictions plus proches des valeurs réelles.



Analyse de Graphe

Déetecter les inefficacités dans la planification des itinéraires et la synchronisation des différents modes de transport.

Transit Network Subgraph

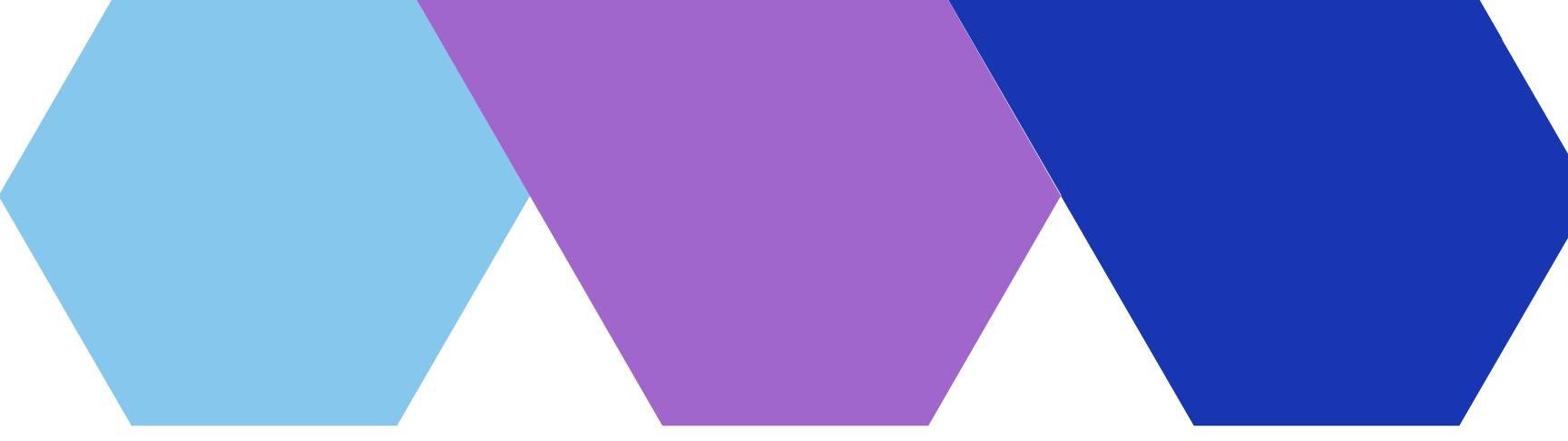


Le nombre d'entrées (In) :

indique combien de connexions mènent vers le nœud.

Le nombre de sorties (Out) :

indique combien de connexions partent de ce nœud.



Ce graphe orienté illustre la structure d'un sous-réseau de transport, où chaque nœud représente une localisation avec ses flux d'entrée et de sortie, et les arêtes modélisent les connexions directionnelles entre les points stratégiques du réseau.

Nœuds (sommets) : Chaque nœud représente un lieu du réseau de transport, identifiés par des étiquettes telles que :

- Origin (Origine)
- Destination

Les nœuds sont indiqués avec des cercles verts, et chaque cercle contient des informations sur le nombre d'entrées (In) et de sorties (Out) pour ce nœud.

Arêtes (lignes) : Les arêtes représentent les connexions ou trajets entre les nœuds. Certaines connexions sont bidirectionnelles, tandis que d'autres sont unidirectionnelles (indiquées par des flèches).

```
# Find specific motifs in the graph  
motifs = g.find("(a)-[e]->(b); (b)-[f]->(c)")  
motifs.show()
```

```
# Shortest path between 'Origin' and 'Tupasaari'  
result = g.bfs(  
    fromExpr="id = 'Origin'",  
    toExpr="id = 'Tupasaari'"  
)  
result.show()
```

```
# Find all trips with mode 'WALK'  
bus_trips = g.find("(a)-[e]->(b)").filter("e.mode == 'WALK'")  
bus_trips.show()
```



Futur plan stratégique

Analyse de l'impact écologique du transport public





Merci pour votre attention

Israa & Sondes