Faculty of Mathematics and Information Science

WARSAW UNIVERSITY OF TECHNOLOGY

Projekt Big Data: Analiza prędkości pojazdów transportu publicznego w Warszawie

Dokumentacja końcowa

Autorzy:
Kacper Grzymkowski
Paulina Jaszczuk
Jakub Fołtyn

Styczeń 2023

Spis treści

1 Wstęp

Niniejszy dokument zawiera dokładne omówienie projektu realizowanego w ramach przedmiotu Big Data realizowanego w trakcie studiów inżynierskich na wydziale Matematyki i Nauk Informacyjnych Politechniki Warszawskiej. W dokumencie zawarto zarówno cel biznesowy i potencjalne korzyści wynikające z realizacji projektu, jak i jego opisy bardziej techniczne: obejmujące architekturę rozwiązania (z uwzględnieniem wykorzystywanych narzędzi), opis przetwarzanych danych jak i sposobów ich pozyskiwania, a także opisy analiz wraz z przykładami. Praca ta zostało ponadto uzupełniona o krótkie podsumowanie projektu oraz podział pracy wśród autorów.

2 Cel projektu

Niniejszy projekt ma za zadanie przedstawienie całościowego procesu pozyskiwania, przetwarzania, składowania oraz dalszych analiz na danych dotyczących autobusów warszawskiego transportu miejskiego oraz danych pogodowych (dotyczących Warszawy). Połączenie owych danych pozwala na wiele różnych szczegółowych analiz, m.in. dotyczących prędkości pojazdów transportu miejskiego w zależności od warunków pogodowych, co jest istotną kwestią bezpieczeństwa pasażerów, może też nieść cenne informacje dla centrali zarządu transportu miejskiego, m.in. o tym, czy kierowcy poszczególnych pojazdów zachowują należytą ostrożność podczas niesprzyjających warunków atmosferycznych.
Naturalnie możliwych analiz jest znacznie więcej, potencjalnie można by też np. kontrolować liczność pojazdów danych linii na danych trasach w zależności od pory dnia czy daty, tym samym sprawdzając skuteczność i "obłożenie" danych tras.

3 Opis danych

Podstawą naszego projektu są dane opisujące obecne położenie autobusów warszawskiego transportu miejskiego. Najważniejszym źródłem owych danych jest strona *Dane po warszawsku*, będąca prostym API pozwalającym na uzyskanie danych dotyczących autobusów warszawskiego transportu miejskiego. Wykonując proste zapytanie w formie adresu URL użytkownik jest w stanie uzyskać dane w formacie JSON obejmujące bieżącą datę. Dane w formacie "surowym" zawierają następujące kolumny:

- **Lat** – współrzędna szerokości geograficznej
- **Lon** – współrzędna długości geograficznej
- **Time** – czas wysłania sygnału GPS pojazdu
- **Lines** – numer linii autobusowej
- **Brigade** – numer brygady pojazdu

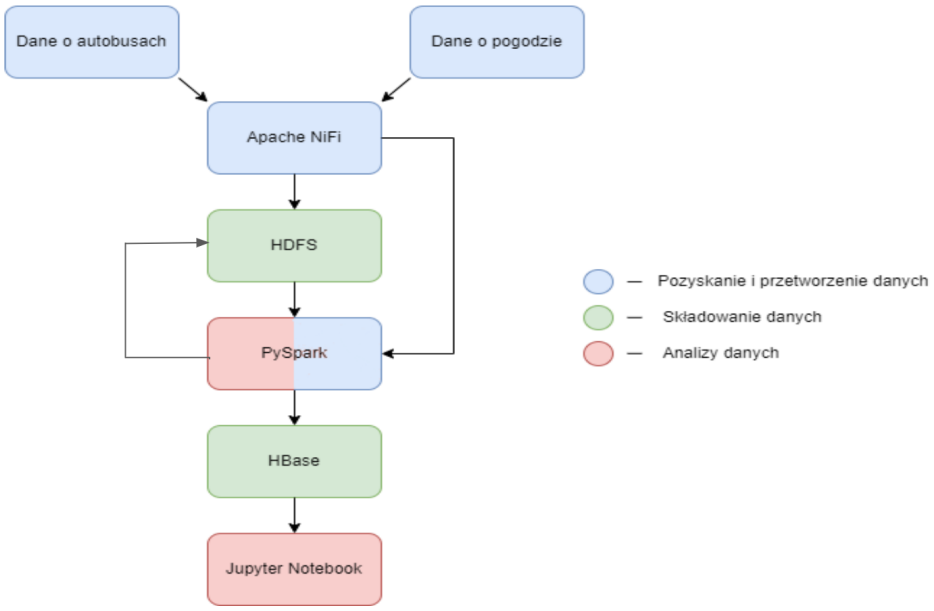
Taki układ zmiennych niesie za sobą wiele cennych informacji, i pozwala na uzyskanie ważnych informacji pochodnych, takich jak np. aktualna prędkość pojazdu, opis uzyskiwania której znajduje się w sekcji 6.
Ponadto niniejszy projekt wzbogacony został o dane pogodowe, pochodzące ze strony *Meteostat*. Ów strona pozwala na pobieranie statystyk pogodowych ze stacji niemal z całego świata- w tym, m.in. z Warszawy. Dane te zapisywane są co godzinę, a historia ich składowania sięga aż lat 20 XX wieku. Połączenie zgodnego czasu oraz lokalizacji pozwala na skuteczne sprawdzenie wpływu obecnych (jak i przeszłych) warunków pogodowych na prędkość poruszania się jak i ogólne zaburzenia w sposobie przemieszczania się autobusów ZTM. Dane w stanie "surowym" posiadają następujące kolumny:

- **date** – data pomiaru.
- **hour** – godzina pomiaru.
- **temp** – temperatura powietrza w stopniach Celsjusza.
- **dwpt** – temperatura punktu rosy w stopniach Celsjusza.
- **rhum** – względna wilgotność powietrza w procentach.
- **prcp** – opady atmosferyczne w mm.
- **snow** – opady śniegu w milimetrach.
- **wdir** – średni kierunek wiatru w stopniach.
- **wspd** – średnia prędkość wiatru.
- **wpgt** – największa prędkosc wiatru.
- **pres** – średnie ciśnienie powietrza.
- **tsun** – całkowite oświetlenie słoneczne w minutach.
- **coco** – kod jakości warunków pogodowych.

Taka konfiguracja zmiennych pozwala na dokładną analizę, biorącą pod uwagę różne czynniki pogodowe.
Warto na koniec zaznaczyć, że w danych tych występują liczne braki w różnych wartościach pomiarów.

4 Architektura rozwiązania

Całość architektury rozwiązania niniejszego projektu, z wyróżnieniem narzędzi wykorzystanych do posczególnych procesów, przedstawiona została na poniższym diagramie:



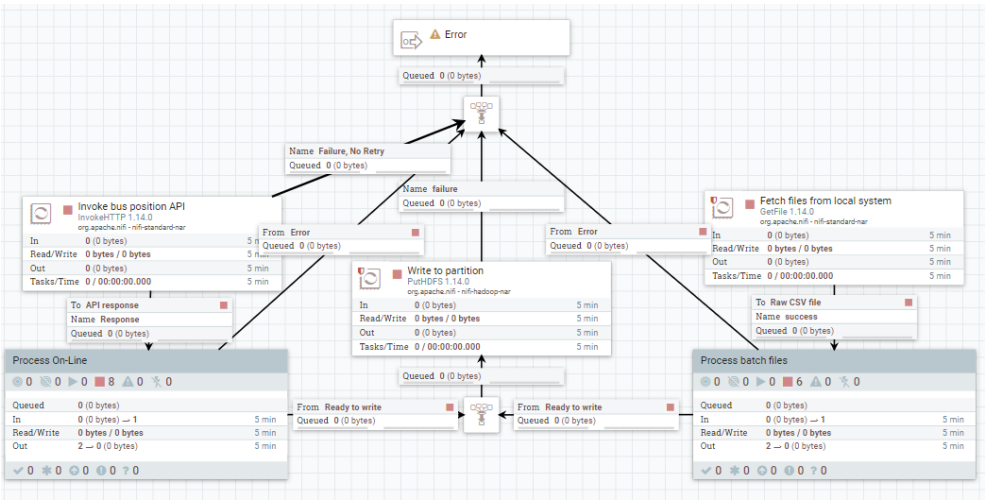
Rysunek 1: Diagram przedstawiający schemat całościowy architektury

Ów diagram, jak i cały projekt, można podzielić na 3 równoważne, i przede wszystkim ściśle związane ze sobą sekcje: **Pozyskanie i przetwarzanie danych**, **Składowanie danych** oraz **Analizy**.
Jak widać na początku schematu, nasze dane pochodzą z 2 różnych źródeł, i dotyczą 2 różnych dziedzin (autobusów transportu miejskiego oraz pogody). Obydwa te rodzaje danych są pozyskiwane w sposób automatyczny, oraz wstępnie przetwarzane w narzędziu Apache Nifi. Następnie dane te są zapisywane do systemu HDFS, skąd pobierane są już skryptem PySparkowym. Co ważne, ów skrypt jest też wywoływany w sposób automatyczny z poziomu Apache Nifi- co symbolizuje dodatkowa strzałka na schemacie. Warto zauważyć, że w PySparku elementy przetwarzania jak i analizy danych są ze sobą w pewien sposób łączone- w PySparku bowiem następuje ubogacenie danych o dodatkowe kolumny (transformacja bronze -¿ silver), jak i następnie utworzenie na podstawie danych pewnych analiz (silver -¿ gold). Analizy te są następnie składowane w HBase, a stamtąd- sczytywane bezpośrednio do Jupyter Notebook, gdzie na ich podstawie tworzone są wizualizacje.
Całość rozwiązania znajduje się na tym repozytorium.

5 Pozyskiwanie i składowanie danych

5.1 Dane o autobusach

Dane dotyczące autobusów w niniejszym projekcie pozyskiwane są na dwa sposoby. Pierwszym z nich jest odpytywanie opisywanego w poprzedniej sekcji API REST-owego. Pozyskanym w ten sposób plikom JSON dodawany jest atrybut zawierający obecny stempel czasowy, a następnie dane te przetwarzane są do formatu AVRO. API odpytywane jest co 30 sekund, a pobrane w ten sposób pliki są łączone. Drugim sposobem jest sczytanie danych historycznych pobranych wcześniej pobranych i zapisanych do jednego pliku CSV. Wstępne przetwarzanie jest tutaj prostsze niż w przypadku wywołania API, i opiera się przede wszystkim na zmianie sposobu zapisu daty oraz przekształcenie formatu do AVRO. Pliki z obydwu ścieżek są następnie łączone i zapisywane do systemu HDFS. Całość procesu obejmuje poniższe flow Apache Nifi:



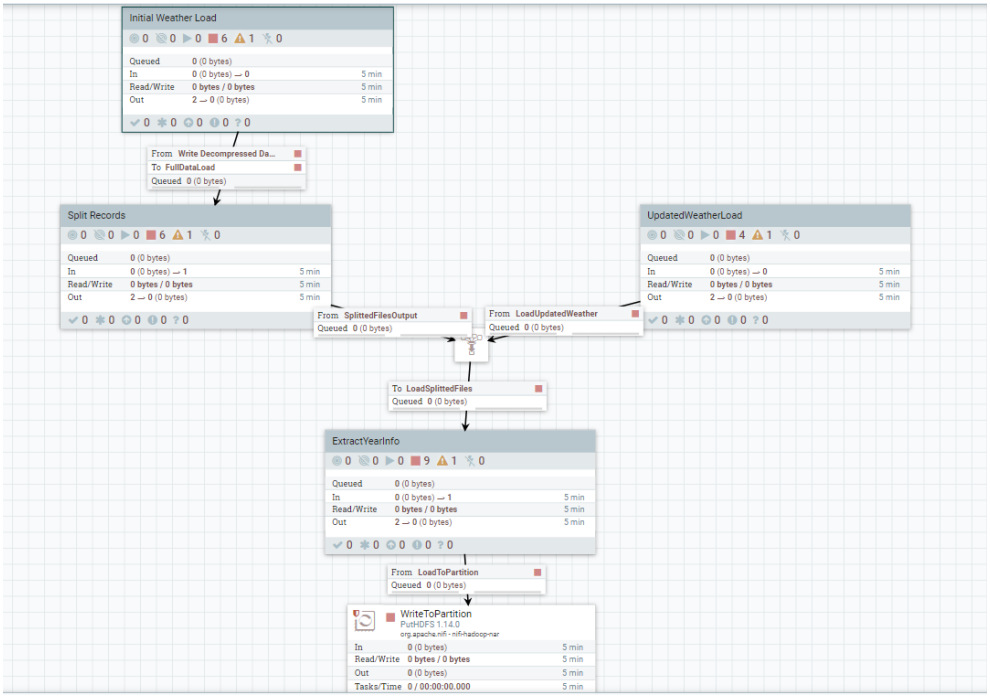
Rysunek 2: Flow w Apache Nifi dla danych o autobusach

- **Process On-Line** – grupa zawierająca wstępne przekształcenia danych pozyskanych z API.
- **Process batch files** – grupa zawierająca wstępne przekształcenia danych pozyskanych z pliku CSV.

5.2 Dane pogodowe

Dane pogodowe, podobnie jak dane dotyczące autobusów, pozyskiwane są na dwa różne sposoby. Pierwszym z nich jest wywołanie podstawowego API strony *Meteostat*, które pozwala na pozyskanie całościowych danych historycznych dla

danej stacji metereologicznej (w naszym przypadku: stacji Warszawa-Okęcie) w formacie CSV. Dane te są pojedynczym plikiem zawierającym cogodzinne rekordy sięgające 1928 roku (warto wspomnieć, że w przypadku najdawniejszych danych pomiary następowały raz dziennie, o godzinie 6). By zasymulować przypływ nowych danych, wywoływane jest inne API, za pośrednictwem skryptu języka Python, które to pozwala na pobranie danych (również w formacie CSV) jedynie z zeszłego tygodnia. Docelowo symulacja "napływu danych" powinna odbywać się raz na tydzień, dlatego też ów skrypt pobiera dane jedynie z takiego okienka czasowego.
Dane z obu źródeł są następnie przetwarzane w jednolity sposób: są one dzielone na podstawie roku danego rekordu, a następnie zapisywane (dalej w formacie CSV) do odpowiednich katalogów na HDFS (każdy katalog zawiera plik z danymi tylko z konkretnego roku, będącego jednocześnie nazwą katalogu). By ułatwić rozdzielenie danych oraz późniejszą ich filtrację, do pliku dodana została kolumna z rokiem obserwacji danego rekordu. Warto tutaj zaznaczyć, że dane historyczne (bezpośrednio z API REST-owego) pobierane są tylko raz, a następnie wywoływany jest skrypt pythonowy, który dopisuje nowe dane. Całość przetwarzania zawarta jest w niniejszym flow Apache nifi:



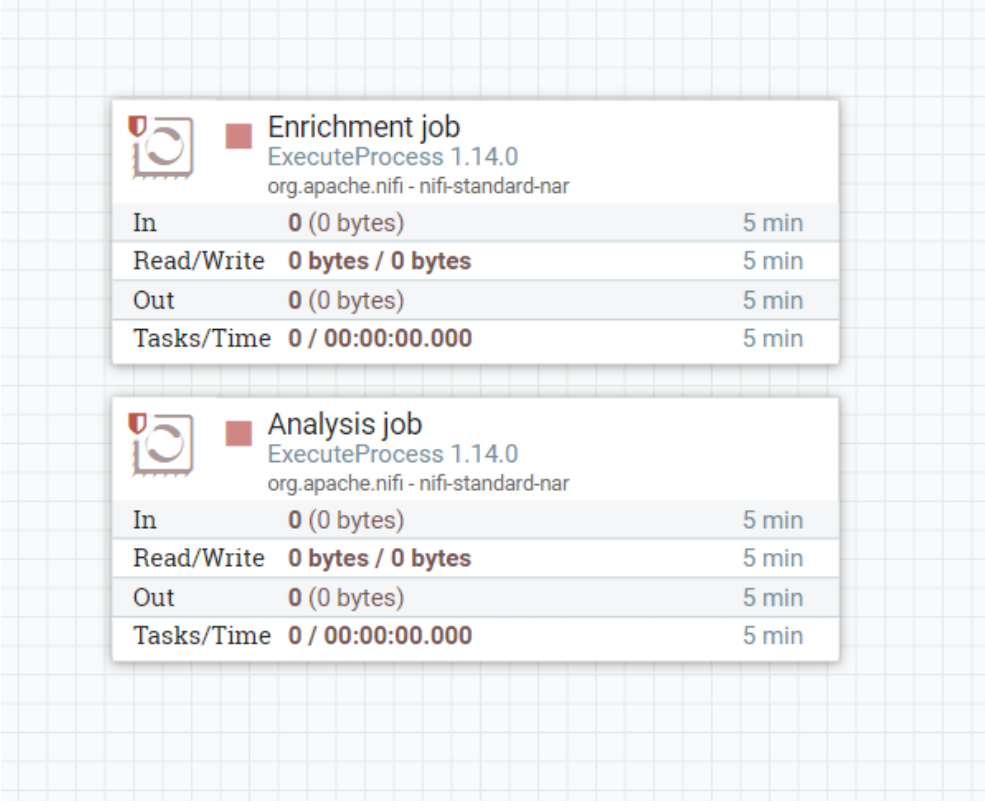
Rysunek 3: Flow w Apache Nifi dla danych pogodowych

- **Initial Weather Load** – grupa zawierająca pobieranie danych historycznych.

- **Split Records** – grupa, w której dane historyczne są wstępnie dzielone na mniejsze pliki.
- **Updated Weather Load** – grupa zawierająca pobieranie danych za pomocą skryptu pythonowego.
- **Extract Year Info** – grupa pozyskująca informacje o roku pozyskania danych rekordów, dzieląca pliki według owych lat.

5.3 Automatyzacja PySpark

Warto wspomnieć o jeszcze jednym schemacie Nifi. Składa się on jedynie z dwóch procesorów



Rysunek 4: Automatyzacja skryptów PySpark w Nifi

Procesory te, dzięki ustawieniu rozkładu uruchamiania komendami z CronTab, odpowiadają za automatyczne uruchomianie skryptów ubogacających oraz przygotowujących analizy z wykorzystaniem Pyspark- tak jak opisano to w następnych sekcjach niniejszego dokumentu.

6 Wzbogacanie danych

Dane przetwarzane w Apache NiFi w ramach naszego rozwiązania są minimalnie przetwarzane, tak aby zachować jak najwięcej oryginalnych informacji. Takie dane jednak nie są wygodne do dalszego przetwarzania, dlatego stworzyliśmy proces wzbogacający dane o dodatkowe informacje, takie jak znalezioną najnowszą poprzednią obserwację, różnicę pozycji i czasu pomiędzy nimi, dystans oraz prędkość chwilowa.

Ta część systemu została zaimplementowana w technologii PySpark, która udostępnia funkcjonalności potrzebne do przeprowadzenia takiego przetwarzania. Program ma jeden parametr: datę dla której ma zostać uruchomione przetwarzanie. Przetwarzanie jest zlecane z poziomu Apache NiFi codziennie, ale może również zostać zlecone manualnie z wykorzystaniem interfejsu konsolowego. Domyślnie system zapisuje efekt wzbogacania danych w przewidywalnym miejscu, ale nie będzie nadpisywać informacji, zgodnie z zaleceniami do tworzenia systemów klasy Big Data. W przypadku konfliktu, program zapisze efekty transformacji w zapasowej lokalizacji pod unikatową nazwą. Proces odbywa się w kilku krokach:

- Ładowanie i zebranie danych w formacie AVRO.
- Inferencja poprzednich obserwacji.
- Obliczanie dystansu, czasu i prędkości pomiędzy obserwacjami.
- Zapisywanie danych w formacie AVRO.

Inferencja poprzednich obserwacji jest wykonywana poprzez podzielenie wszystkich danych po numerze pojazdu, posortowanie podzielonych danych według czasu otrzymanego w obserwacji oraz przesunięcie kolumn o jedną. Taka operacja prawidłowo znajduje poprzednią obserwację przy założeniu, że ta obserwacja istnieje w tej samej ramce danych oraz numery pojazdów są unikatowe. Na podstawie informacji o poprzedniej obserwacji jest obliczany czas pomiędzy tymi obserwacjami oraz dystans. Dystans jest obliczany przybliżoną metodą odwzorowania walcowego równoodległościowego. Obliczanie odległości tą metodą jest opisane następującymi wzorami:

$$
\begin{aligned}
x &= \Delta\lambda \cdot \cos((\phi_1 + \phi_2)/2) \\
y &= \Delta\phi \\
d &= R \cdot \sqrt{x^2 + y^2},
\end{aligned}
$$

gdzie R to promień Ziemi, ϕ to szerokość geograficzna w radianach, a λ to długość geograficzna w radianach.

7 Opis przeprowadzanych analiz

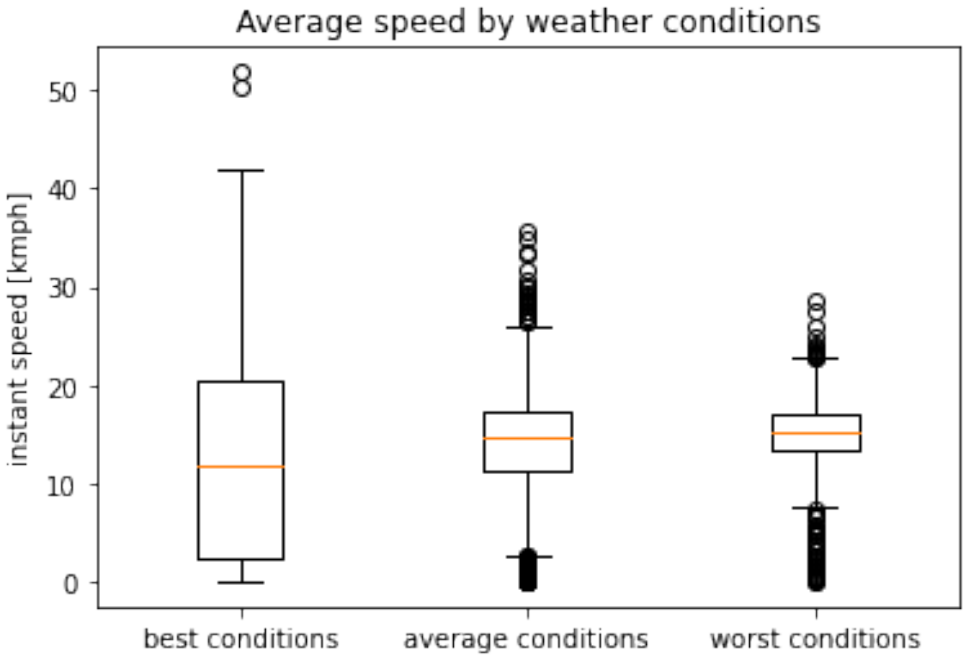
Wzbogacone uprzednio dane poddano następnie analizom przy pomocy technologii PySpark. Dokonano grupowania danych po liniach lub konkretnych numerach pojazdów oraz obliczono następujące statystyki.

- Łączny przejechany dystans, czas w trasie oraz średnie prędkości dla pojazdów.
- Łączny przejechany dystans, czas w trasie oraz średnie prędkości dla linii.
- Łączny przejechany dystans, czas w trasie oraz średnie prędkości wszystkich pojazdów w zależności od kondycji pogodowej
- Łączny przejechany dystans, czas w trasie oraz średnie prędkości linii w zależności od ilości opadów
- Łączny przejechany dystans, czas w trasie oraz średnie prędkości pojazdów w zależności od prędkości wiatru.

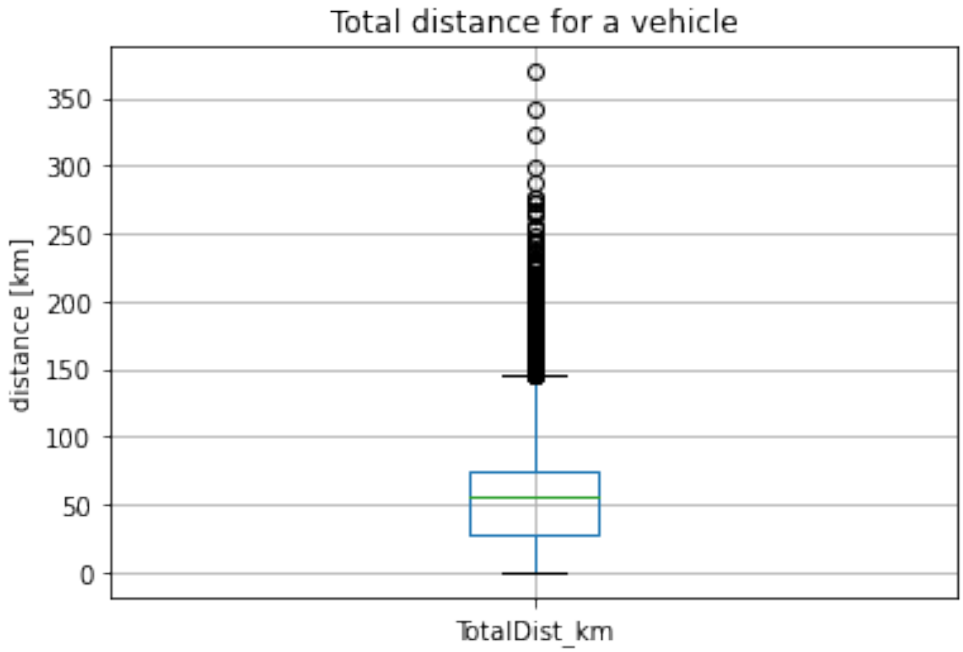
Z powodu małej różnorodności danych w okresie, na którym wykonywaliśmy analizy, dane dotyczące prędkości wiatru i deszczu zostały zagregowane do dwóch wartości - 0 i 1, gdzie 0 oznacza kolejno słaby wiatr (poniżej 15km/h) oraz brak opadów, a 1 pozostałe wartości. Uzyskane raporty zapisano zostały zapisane do Apache HBase. Skrypt wywoływany jest manualnie z poziomu konsoli.

8 Przykładowy konsument analiz

Zagregowane i poddane wcześniejszej analizie dane zostały przykładowo zwizualizowane w końcowym konsumencie (Jupyter notebook). Na wizualizacjach przedstawiono dane z 31.12.2022r. Skupiają się one na wspomnianych uprzednio obszarach - bezpieczeństwie podróżnych oraz podejmowaniu decyzji drogowych przez zarząd transportu miejskiego. Jak widać na wykresie 5 warunki pogodowe mają istotny wpływ na średnią prędkość poruszania się pojazdów. Zaskakujące może być to, że nie jest ona niższa, lecz mniej zróżnicowana. Z wykresu 6 możemy dowiedzieć się, jak wygląda rozkład pokonywanego dziennie dystansu przez pojazdy danej linii, co może skłonić zarząd do dodania większej liczby autobusów do danych połączeń.



Rysunek 5: Średnia prędkość linii w zależności od warunków pogodowych.



Rysunek 6: Łączny przejechany dystans dla poszczególnych pojazdów w ciągu doby.

9 Testy

CEL TESTU	METODA	OCZEKIWANY WYNIK	FAKTYCZNY WYNIKI
Sprawdzenie wstępnego ładowania danych pogodowych	Uruchomienie flow "wstępnego" w Nifi i potwierdzenie utworzenia folderów	W odpowiednim folderze utworzone podfoldery z danymi podzielonymi na lata	7, 8
Sprawdzenie uzupełnienia danych pogodowych	Uruchomienie flow w Nifi symulującego deltę i sprawdzenie porównanie liczby rekordów folderów "2022" oraz "2023"	Pliki CSV w folderach "2022" oraz "2023" powiększyły się rekordy z ostatnich dni	9, 10 oraz 11, 12
Sprawdzenie ładowania danych o autobusach z pliku	Uruchomienie flow w Nifi ładującego dane z pliku	Plik załadowany do HDFS w formacie avro	13
Sprawdzenie ładowania danych o autobusach z API	Uruchomienie flow w Nifi ładującego dane z API	Plik załadowany do HDFS w formacie avro w folderze z odpowiednią datą	14
Sprawdzenie czy analizy zostały utworzone	Wyświetlenie powstałych ramek danych w konsoli	Ramki danych wyświetlone w konsoli	15, 16, 17, 18 oraz 19

9.1 Wyniki testów

```
drwxr-xr-x   - root supergroup          0 2023-01-05 13:52 /user/kpk/weather/201
2
drwxr-xr-x   - root supergroup          0 2023-01-05 13:54 /user/kpk/weather/201
3
drwxr-xr-x   - root supergroup          0 2023-01-05 13:54 /user/kpk/weather/201
4
drwxr-xr-x   - root supergroup          0 2023-01-05 13:56 /user/kpk/weather/201
5
drwxr-xr-x   - root supergroup          0 2023-01-05 13:57 /user/kpk/weather/201
6
drwxr-xr-x   - root supergroup          0 2023-01-05 13:59 /user/kpk/weather/201
7
drwxr-xr-x   - root supergroup          0 2023-01-05 13:58 /user/kpk/weather/201
8
drwxr-xr-x   - root supergroup          0 2023-01-05 13:58 /user/kpk/weather/201
9
drwxr-xr-x   - root supergroup          0 2023-01-05 13:59 /user/kpk/weather/202
0
drwxr-xr-x   - root supergroup          0 2023-01-05 13:59 /user/kpk/weather/202
1
drwxr-xr-x   - root supergroup          0 2023-01-05 14:00 /user/kpk/weather/202
2
drwxr-xr-x   - root supergroup          0 2023-01-05 14:00 /user/kpk/weather/202
3
vagrant@node1:~$ _
```

Rysunek 7: Wynik komendy hadoop fs -ls /user/kpk/weather (dowód powstania folderów)

```
vagrant@node1:~$ hadoop fs -ls /user/kpk/weather | wc -l
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.7.6/share/hadoop/common/li
b/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/apache-tez-0.9.1-bin/lib/slf4j-log4
j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
77
```

Rysunek 8: Całkowita liczba powstałych folderów- 77- zgadza się z liczbą zarejestrowanych lat w pliku- od 1928r (nastąpiła przerwa w mierzeniu spowodowana 2 Wojną Światową)

```
vagrant@node1:~$ hadoop fs -cat /user/kpk/weather/2022/2022.csv | wc -l
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.7.6/share/hadoop/common/li
b/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/apache-tez-0.9.1-bin/lib/slf4j-log4
j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
8770
```

Rysunek 9: Liczba rekordów w pliku 2022.csv przed uruchominiem flow uzupełniającego- 8770

```
vagrant@node1:~$ hadoop fs -cat /user/kpk/weather/2022/2022.csv | wc -l
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.7.6/share/hadoop/common/li
b/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/apache-tez-0.9.1-bin/lib/slf4j-log4
j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
8818
```

Rysunek 10: Liczba rekordów w pliku 2022.csv po uruchomieniu flow uzupełniającego- 8818

```
vagrant@node1:~$ hadoop fs -cat /user/kpk/weather/2023/2023.csv | wc -l
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.7.6/share/hadoop/common/li
b/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/apache-tez-0.9.1-bin/lib/slf4j-log4
j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
162
```

Rysunek 11: Liczba rekordów w pliku 2023.csv przed uruchominiem flow uzupełniającego- 162

```
vagrant@node1:~$ hadoop fs -cat /user/kpk/weather/2023/2023.csv | wc -l
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.7.6/share/hadoop/common/li
b/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/apache-tez-0.9.1-bin/lib/slf4j-log4
j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
492
```

Rysunek 12: Liczba rekordów w pliku 2023.csv po uruchomieniu flow uzupełniającego- 492

```
Starting standalonesession daemon on host node1.
Starting taskexecutor daemon on host node1.
vagrant@node1:~$ hadoop fs -ls /user/kpk/bus
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.7.6/share/hadoop/common/li
b/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/apache-tez-0.9.1-bin/lib/slf4j-log4
j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Found 1 items
drwxr-xr-x   - root supergroup          0 2023-01-06 11:10 /user/kpk/bus/2022100
7
vagrant@node1:~$ hadoop fs -ls /user/kpk/bus/20221007
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.7.6/share/hadoop/common/li
b/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/apache-tez-0.9.1-bin/lib/slf4j-log4
j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Found 1 items
-rw-r--r--   1 root supergroup   38607394 2023-01-06 11:10 /user/kpk/bus/2022100
7/batch_20230106_110958.avro
vagrant@node1:~$ _
```

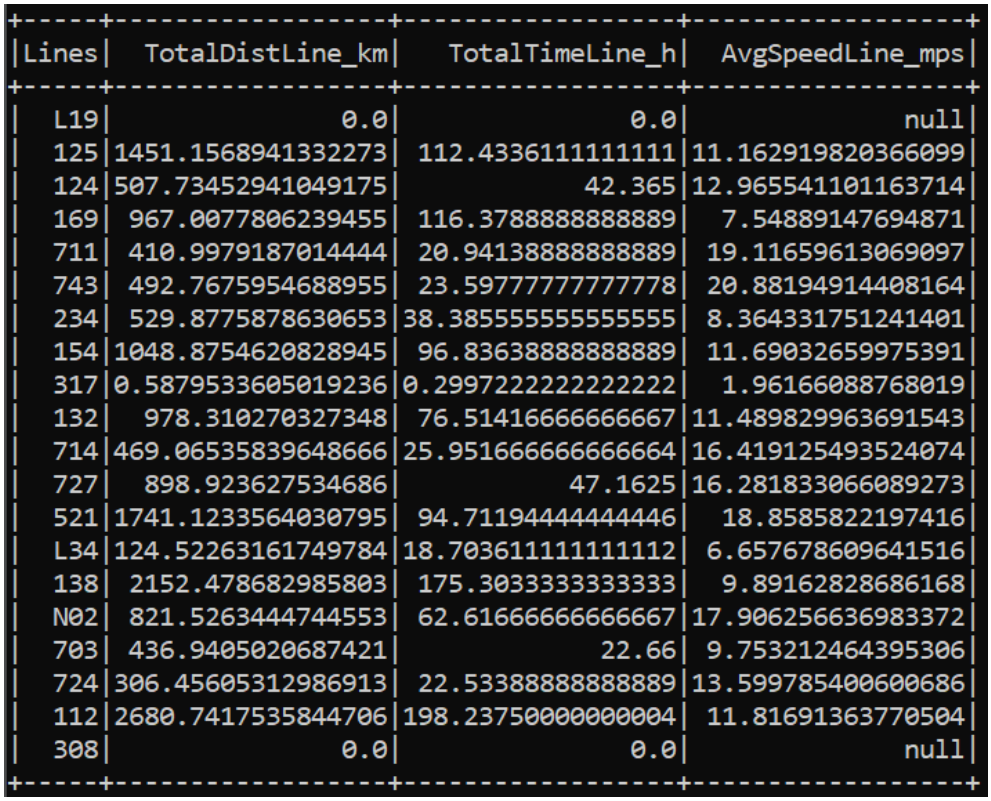
Rysunek 13: Plik załadowany w formacie avro w folderze z odpowiednią datą (dane z pliku pochodzą z 07.2022)

```
vagrant@node1:~$ hadoop fs -ls /user/kpk/bus
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.7.6/share/hadoop/common/li
b/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/apache-tez-0.9.1-bin/lib/slf4j-log4
j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Found 1 items
drwxr-xr-x   - root supergroup          0 2023-01-06 11:42 /user/kpk/bus/2023010
6
vagrant@node1:~$ hadoop fs -ls /user/kpk/bus/20230106
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.7.6/share/hadoop/common/li
b/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/apache-tez-0.9.1-bin/lib/slf4j-log4
j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Found 1 items
-rw-r--r--   1 root supergroup    2442483 2023-01-06 11:42 /user/kpk/bus/2023010
6/streamed_20230106_123845.avro
vagrant@node1:~$
```

Rysunek 14: Plik załadowany w formacie avro w folderze z odpowiednią datą (dane uzyskane zostały dnia 06.01.2022)

Lines	VehicleNumber	Brigade	TotalDist_km	TotalTime_h	AvgSpeed_kmph
156	9357	3	184.9503642988991	19.668333333333333	9.403458908511098
239	9248	51	94.42536818385379	6.079166666666667	15.532617110435169
138	5440	7	265.4065675028305	22.558888888888887	11.765054955058242
707	9561	1	280.14582013400985	15.020555555555555	18.650829464852524
190	9942	8	280.28003858707183	17.884722222222223	15.67147843307383
210	6229	1	338.4222540091434	18.038888888888888	18.760703948766803
750	777		8.127388671564512	0.7794444444444445	10.42715581526452
189	8823	8	0.0	0.0	null
167	2248	9	0.0	0.0	null
107	9205	4	271.69434721964825	21.9375	12.384927508587955
715	9551	52	169.47039914161127	7.598888888888889	22.30199725507386
123	1540	09	0.0	0.0	null
L39	70002	3	0.0	0.0	null
716	2241	2	0.022520311277957495	0.10972222222222222	0.2052484065839164
112	3464	7	266.3074504832456	16.96361111111111	15.698747674592415
523	8833	16	0.0	0.0	null
133	1041	1	0.0	0.0	null
713	9553	65	2.2394664904999497	0.10972222222222222	20.410327508353973
211	9080	1	248.35274222198527	19.373055555555556	12.819492594226617
193	9954	2	191.80952160370325	17.558055555555555	10.924303149445992

Rysunek 15: Łączny przejechany dystans, czas w trasie oraz średnie prędkości dla pojazdów.



Lines	TotalDistLine_km	TotalTimeLine_h	AvgSpeedLine_mps
L19	0.0	0.0	null
125	1451.1568941332273	112.4336111111111	11.162919820366099
124	507.73452941049175	42.365	12.965541101163714
169	967.0077806239455	116.3788888888889	7.54889147694871
711	410.9979187014444	20.94138888888889	19.11659613069097
743	492.7675954688955	23.59777777777778	20.88194914408164
234	529.8775878630653	38.385555555555555	8.364331751241401
154	1048.8754620828945	96.83638888888889	11.69032659975391
317	0.5879533605019236	0.2997222222222222	1.96166088768019
132	978.310270327348	76.51416666666667	11.489829963691543
714	469.06535839648666	25.951666666666664	16.419125493524074
727	898.923627534686	47.1625	16.281833066089273
521	1741.1233564030795	94.71194444444446	18.8585822197416
L34	124.52263161749784	18.703611111111112	6.657678609641516
138	2152.478682985803	175.3033333333333	9.89162828686168
N02	821.5263444744553	62.61666666666667	17.906256636983372
703	436.9405020687421	22.66	9.753212464395306
724	306.45605312986913	22.53388888888889	13.599785400600686
112	2680.7417535844706	198.23750000000004	11.81691363770504
308	0.0	0.0	null

Rysunek 16: Łączny przejechany dystans, czas w trasie oraz średnie prędkości dla linii.

Lines	VehicleNumber	Brigade	coco	TotalDist_km	TotalTime_h	AvgSpeed_kmph
187	7328	2	7	67.00308632260538	4.0008333333333335	16.747282563450625
165	8328	4	4	43.554971833475044	4.001388888888889	10.884963457168354
N11	1554	122	3	17.945230136505526	0.9961111111111111	18.015289596045704
190	9944	10	4	63.1339578769341	4.003611111111111	15.769253337748058
117	9422	2	4	51.43967715746008	4.013888888888889	12.81542129874438
120	5403	4	7	69.69345500216983	3.9919444444444445	17.45852327658558
210	5436	2	3	183.72654638332617	14.855277777777777	12.367762429738294
212	9693	3	7	61.77887094365327	4.0	15.444717735913317
L18	60051	M2	3	1.3937791616509752	0.04861111111111111	28.67202846824863
107	9204	5	4	58.79463058150969	4.313611111111111	13.630025764275542
147	1523	59	3	104.56587471221378	5.4725	19.107514794374378
157	9418	6	7	49.267433629259315	3.993611111111111	12.336562639308168
N43	5987	3	2	48.82568065951404	2.9925	16.31601692882675
523	5245	4	7	67.31724233463324	3.9908333333333332	16.867966339853808
521	9826	3	7	71.13100508637183	3.991388888888889	17.82111617446855
146	1521	55	7	22.093296870128455	1.6697222222222223	13.231719968800936
142	9338	61	4	68.9851784943944	3.9966666666666666	17.26067852236724
714	9518	53	4	25.781585382428165	1.586111111111111	16.25458973322967
164	9231	1	7	60.819519311764275	4.001111111111111	15.200657423101317
527	3403	54	3	55.30438879554803	3.5869444444444443	15.418245153254311

Rysunek 17: Łączny przejechany dystans, czas w trasie oraz średnie prędkości wszystkich pojazdów w zależności od kondycji pogodowej.

Lines	Rain	TotalDistLine_km	AvgDistLine_km	AvgTimeLine_h	AvgSpeedLine_mps
N63	0	259.1527127640668	51.830542552813355	2.100333333333333	24.312171652158863
209	0	680.1446136143858	170.03615340359644	12.724652777777777	13.351884917043296
184	0	1520.1695154176095	138.19722867432813	12.751489898989899	7.632749130908655
704	0	730.8549932031303	146.17099864062607	7.397722222222223	16.890344565050036
511	0	1125.4845220821098	140.68556526026373	9.809305555555556	13.452064341446109
509	0	3196.445884951621	152.21170880722005	10.530079365079365	13.029197643797412
514	0	1765.2384032844677	176.52384032844677	12.304944444444445	11.119264488724598
N38	0	113.57997993815499	28.394994984538748	1.7878472222222221	16.43516623281849
192	0	589.8392232438764	58.98392232438764	5.4352222222222215	10.856362896006468
211	0	1109.2956202660355	184.88260337767258	13.93037037037037	13.21253986171461
207	0	1153.0812617727975	288.27031544319937	18.295625	15.763394098756624
512	0	1246.0554805221213	155.75693506526517	13.001666666666665	10.532749610085036
N24	0	448.26246607857877	64.03749515408268	9.648928571428572	14.615736825502484
153	0	765.1128800046563	191.27822000116407	15.176805555555553	11.036530536236858
186	0	2767.0712962404473	251.55193602185884	13.91510101010101	14.853976186406866
735	0	789.9332582168281	112.84760831668973	8.482261904761904	12.78877966884395
165	0	802.8730492163012	100.35913115203765	13.33184027777778	8.268936825137517
114	0	2069.0664595484536	258.6333074435567	16.930937500000002	16.400032796463304
133	0	677.9385859966308	96.84836942809011	5.806666666666667	11.651822186997968
245	0	624.4341283654076	62.44341283654076	5.538777777777779	12.179962383023888

Rysunek 18: Łączny przejechany dystans, czas w trasie oraz średnie prędkości linii wzależności od ilości opadów.

Lines	VehicleNumber	Brigade	Wind	TotalDist_km	TotalTime_h	AvgSpeed_kmph
N45	2231	195	0	18.67481360987876	1.0822222222222222	17.25598793520625
140	5418	07	0	0.0	0.0	null
125	8834	5	0	296.31011537490383	24.040277777777778	12.325569534342293
116	5919	6	0	0.0	0.0	null
190	9936	3	0	210.8240501814637	13.1675	16.010939827717007
161	9068	M3	0	144.3744511851797	9.955555555555556	14.501897998511355
147	1529	55	0	0.04153811432580767 3	3.8513888888888888	0.010785229828554463
523	7254	5	0	0.0	0.0	null
727	1214	61	0	271.15435101441136	13.376111111111111	20.27153847347844
180	5964	5	0	0.016705339115986092	0.1961111111111112	0.08518303231947581
111	5946	7	0	269.0569474520953	19.50027777777778	13.79759563008423
141	4222	5	0	0.3997574106995629	0.3238888888888889	1.2342424344068836
199	4414	M55	0	92.86116484943302	8.596944444444444	10.801647660924711
N36	8319	371	0	8.628188947205704	0.6208333333333333	13.89775400892194
182	5238	4	0	302.2274501104206	28.73611111111111	10.517339974843056
116	5933	7	0	313.0316120009612	20.29611111111111	15.423231095221587
504	8560	4	0	287.2535997426125	27.50861111111111	10.442315629180811
120	5416	5	0	10.344656729647221	0.8794444444444445	11.76271769637713
N25	6211	193	0	106.03859567501559	4.649444444444445	22.806723887564587
172	8807	8	0	5.321103903881362	0.5947222222222223	8.947208806152686

Rysunek 19: Łączny przejechany dystans, czas w trasie oraz średnie prędkości pojazdów w zależności od prędkości wiatru.

10 Podsumowanie

W dokumentacji przedstawiony został kompletny proces przetwarzania danych od ich pozyskania, do stworzenia raportów i wizualizacji. Z punktu widzenia biznesowego przeprowadzone analizy mogą być cenne zarówno dla zarządu transportu miejskiego jak i jego bezpośrednich użytkowników. Stworzone wizualizacje odpowiadają na pytania postawione przez nas przed realizacją projektu dotyczące bezpieczeństwa podróżnych oraz mogą okazać się pomocne w podejmowaniu decyzji dotyczących funkcjonowania komunikacji publicznej.

11 Podział pracy

Jakub Fołtyn

- Dokumentacja końcowa
- Proces ładujący dane pogodowe w Apache NiFi
- Testy procesów w Apache NiFi

Kacper Grzymkowski

- Proces ładujący dane o autobusach w Apache NiFi
- Proces wzbogacania danych autobusowych
- Dokumentacja końcowa

Paulina Jaszczuk

- Analizy i testy analiz
- wizualizacje w Jupyter notebooku
- Dokumentacja końcowa