Hurtownie i Eksploracja Danych	
Temat	Sieć restauracji
Przygotowali	Piotr Kaczmarczyk, Przemysław Postrach

1. Opis problemu

W ramach projektu opracowaliśmy hurtownie danych wspomagającą prace sieci restauracji “Vege cziken w majo”. Dane pochodzą z systemu transakcji synchronizującego prace całej sieci. Zakupy mogą być dokonywane osobiście w restauracji, w specjalnych kioskach oraz online w aplikacji. Płatności są przyjmowane przez wszystkie dostępne formy min. Gotówkowo, PAYU czy kartą płatniczą. Dzięki dacie wygenerowania paragony przeprowadzana jest synchronizacja w i kolejność dodawania rekordów w bazie.

W systemie rejestrowane są również dostawy. Aplikacja wysyła informację o miejscu dostawy oraz o pracowniku odpowiedzialnym za przygotowanie i skompletowanie zamówienia. W sieci restauracji produkty są pogrupowane w kategorie i zorganizowane w różne menu. Każda restauracja posiada takie same produkty, kategorie oraz menu.

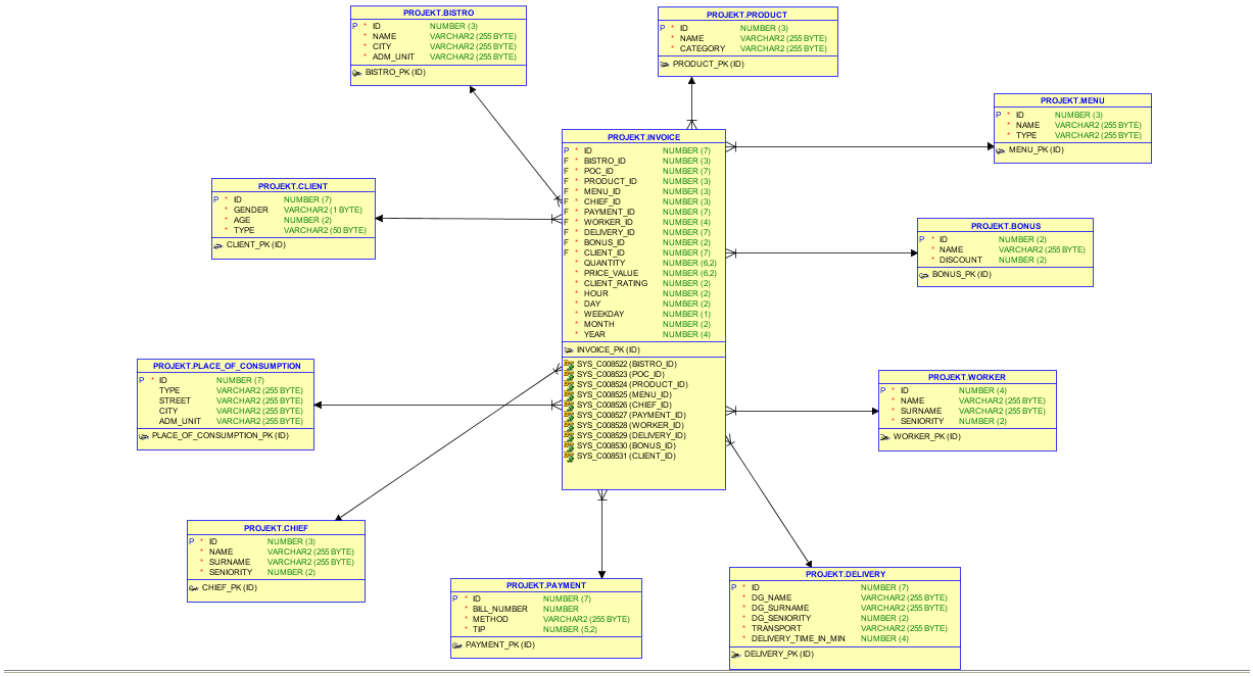
Ponadto sieć oferuje różne bonusy, które można wykorzystać przy zakupach. Dostępne dla wszystkich klientów, nie tylko posiadaczy kart stałego klienta. Z pomocą sztucznej inteligencji system może codziennie generować promocje dla klientów oraz przewidywać jakie promocje będą się najchętniej sprzedawały. Dzięki temu możemy generować popyt i podaż w zależności od stanów magazynowych czy okresów gorszej sprzedaży.

Każdy klient może ocenić każdy posiłek w skali od 1 do 10 przy kasie. Dzięki temu możemy przechowywać informacje o tym, co, kiedy i gdzie klientom podobało się bardziej lub mniej. Dane te są podstawą do analiz danych i predykcji poprzez sieci neuronowe. Najbardziej podstawowym elementem analizy jest sprawdzenie czy dana transakcja ucieszyła klienta. Pozwala to nam przewidzieć co należy poprawić i udoskonalić w naszych restauracjach.

System zarządzania siecią działa w trybie online i na bieżąco dostarcza dane do magazynu. W ramach projektu zaprojektowaliśmy go i symulowaliśmy dane wprowadzane do niego przez system na przestrzeni kilku miesięcy. Hurtowania posiada dane nie tylko o transakcji zakupionych produktach, ale też o pracownikach, klientach, dostępnym menu oraz kiedy transakcja została przeprowadzona.

2. Hurtownia danych

Schemat bazy danych:



Hurtowania danych realizuje klasyczny schemat gwiazdy. Wszystkie tabele posiadają połączenie jeden do wielu z tabelą “Invoice”.

Miary:

1. Quantity – waga zakupionego posiłku
2. Price_value – cena
3. Client_rating – ocena klienta
4. Data

Zostały wybrane ze względu na optymalizacje zapytań oraz możliwości późniejszej analizy data minerem.

Wymiary:

1. Client – informacje z karty stałego klienta
2. Place_of_consumption – informacje o miejscu konsumpcji
3. Bistro – informacje o restauracji
4. Menu – posiada nazwę i typ
5. Bonus – informacje o zniżce
6. Produkt – informacje o produkcie i jego kategorii
7. Worker – informacje o pracowniku
8. Chief – informacje o szefie kuchni
9. Payment – informacje o płatności
10. Delivery – informacje o dostawie

Wymiary pomagają nam zebrać bardziej szczegółowe informacje o restauracji, klientach oraz posiłkach. Dzięki temu profilowanie klientów będzie sprawniejsze. Będziemy mogli sprawdzić jakie oni mają upodobania oraz co najczęściej się sprzedaje.

3. Instalacja

Instalacja oraz konfiguracja bazy danych przebiegłą w skrócony sposób. Wykorzystaliśmy obraz maszyny wirtualnej opartej na WIN 10 z zainstalowanym Oracle sql server oraz sql devleoper. Została ona nam udostępniona przez prowadzącego.

Należało jedynie nadać uprawnienia do dla użytkownika “projekt” aby można było bez przeszkód wykonywać zadania na bazie danych. Zostało to zrobione poprzez egzekucje komendy:

```
GRANT ALL PRIVILEGES TO PROJEKT;
```

Przygotowany obraz posiadał dostępny moduł “Data miner” należało go jedynie pobrać przy pierwszym uruchomieniu bazy. Często można spotkać się z błędami przy jego instalacji, dlatego na postawie miejsca oraz nazwy wykonywanego SQLa należy w odpowiednim miejscu dodać poniższą linijkę:

```
alter session set "_oracle_script" = true;
```

4. Zasilenie hurtowni danymi

Zasilenie hurtowni danymi odbyło się poprzez skrypt zawarty w pliku “createAndLoad.bat”.

```
sqlplus %1 @buildDB.sql
cd ctl
for %%f in (*.ctl) do (
    sqlldr CONTROL=%%f  log=../logs/%%~nf.log bad=../bads/%%~nf.log  skip=1 userid=%1
)
cd..
```

Skrypt odpalamy w konsoli komendom:

```
CreateAndLoad.bat *użytkownik*/*hasło*
```

W pierwszej linii zostaje uruchomiony sql plus i wykonany kod SQL usuwający bazę oraz jej zależności i tworzący tabele na nowo.

Następnie zostają wykonane skrypty ładujące dane do bazy danych. Ze względu na więzy integralności tabela “Invoice” zostaje załadowana na samym końcu. W tym celu nazwa zmieniona została na “ww_Invoice.ctl”.

5. Zapytania SQL

Ustawienia bazy danych spowodowało brak obsługi polskich znaków.

Cube:

1. Zapytanie pokazuje która produkty są najbardziej rentowne w restauracji

```
SELECT
    NVL(TO_CHAR(BIS.NAME), 'FOR ALL') "BISTRO",
    NVL(TO_CHAR(PROD.NAME), 'FOR ALL') "PRODUCT",
    "SALES COUNT",
    TAKINGS
FROM
    (
        SELECT
            BISTRO_ID,
            PRODUCT_ID,
            count(*) AS "SALES COUNT",
            sum(PRICE_VALUE) AS TAKINGS
        FROM
            INVOICE INV
        GROUP BY
            cube (INV.BISTRO_ID, INV.PRODUCT_ID)
        order by
            INV.BISTRO_ID,
            INV.PRODUCT_ID
    ) MAIN
    JOIN BISTRO BIS ON MAIN.BISTRO_ID = BIS.ID
    LEFT OUTER JOIN PRODUCT PROD ON MAIN.PRODUCT_ID = PROD.ID;
```

	BISTRO	PRODUCT	SALES COUNT	TAKINGS
1	Szybki Dom	Tea	949	33670,73
2	Szybki Dom	Water	882	31019,51
3	Szybki Dom	Mirinda	960	33392,24
4	Szybki Dom	Coffie	921	31553,9
5	Szybki Dom	Grass salad	892	31700,96
6	Szybki Dom	Greek salad	895	31731,98
7	Szybki Dom	Caesar salad	918	32604,92
8	Szybki Dom	Salmon salad	886	31585,29
9	Szybki Dom	Herring salad	930	32524,22
10	Szybki Dom	Mushroom soup	887	31499,85
11	Szybki Dom	Pappa al pomodoro	928	31573,22
12	Szybki Dom	Asparagus soup	898	31905,94
13	Szybki Dom	Strogonov	853	29864,89
14	Szybki Dom	Schabowy	910	32069,78
15	Szybki Dom	Big beef	948	33661,28

2. Zapytanie pokazuje najczęściej występujący bonus w danej restauracji

```sql
SELECT
    NVL(TO_CHAR(BON.NAME), 'ALL') AS BONUS,
    NVL(TO_CHAR(BIS.NAME), 'ALL') AS BISTRO,
    "SALES COUNT"
FROM
    (
        SELECT
            BONUS_ID,
            BISTRO_ID,
            COUNT(*) AS "SALES COUNT"
        FROM
            INVOICE INV
        GROUP BY
            CUBE (INV.BONUS_ID, INV.BISTRO_ID)
        ORDER BY
            INV.BONUS_ID ASC,
            INV.BISTRO_ID ASC
    ) MAIN
    JOIN BONUS BON ON BON.ID = MAIN.BONUS_ID
    LEFT OUTER JOIN BISTRO BIS ON MAIN.BISTRO_ID = BIS.ID;
```

	BONUS	BISTRO	SALES COUNT
1	None	Szybki Dom	4025
2	For student	Szybki Dom	4045
3	For workers	Szybki Dom	4045
4	Big family	Szybki Dom	4015
5	Usmiech bombelka	Szybki Dom	3961
6	None	Wielki Talerz	3878
7	For student	Wielki Talerz	3891
8	For workers	Wielki Talerz	4029
9	Big family	Wielki Talerz	4017
10	Usmiech bombelka	Wielki Talerz	4016
11	None	Wielki Kogut	3886
12	For student	Wielki Kogut	3982
13	For workers	Wielki Kogut	3934
14	Big family	Wielki Kogut	4022

3. Zapytanie pokazuje jakie menu jest najczęściej wybierane w danej restauracji.

```
SELECT
    NVL(TO_CHAR(MEN.NAME), 'ALL') AS MENU,
    NVL(TO_CHAR(BIS.NAME), 'ALL') AS BISTRO,
    NVL(TO_CHAR(MAIN.CLIENT_RATING), 'ALL') as RATING,
    SALES_COUNT AS "SALES COUNT"
FROM
    (
        SELECT
            CLIENT_RATING,
            MENU_ID,
            BISTRO_ID,
            COUNT (*) SALES_COUNT
        FROM
            INVOICE INV
        GROUP BY
            cube (INV.MENU_ID, INV.CLIENT_RATING, INV.BISTRO_ID)
        ORDER BY
            INV.CLIENT_RATING ASC,
            INV.CLIENT_RATING ASC,
            INV.BISTRO_ID ASC
    ) MAIN
    JOIN MENU MEN  ON MAIN.MENU_ID = MEN.ID
    LEFT OUTER JOIN BISTRO BIS ON MAIN.BISTRO_ID = BIS.ID;
```

	MENU	BISTRO	RATING	SALES COUNT
1	Teges Weges	Szybki Dom	1	271
2	Teges Weges	Wielki Talerz	1	250
3	Teges Weges	Wielki Kogut	1	242
4	Teges Weges	Wspanialy Banan	1	267
5	Teges Weges	Wspanialy Bochenek	1	286
6	Teges Weges	Swiezy Bochenek	1	302
7	Teges Weges	Wielki Schab	1	299
8	Teges Weges	Szybki Obiad	1	271
9	Teges Weges	Wielki Dom	1	273
10	Teges Weges	Swiezy Dom	1	242
11	Teges Weges	ALL	1	2703
12	Teges Weges	Szybki Dom	2	268
13	Teges Weges	Wielki Talerz	2	279
14	Teges Weges	Wielki Kogut	2	272
15	Teges Weges	Wspanialy Banan	2	259

Grouping sets:

1. Zapytanie pokazuje kucharza, wykonany posiłek, ocenę klienta oraz ilość przygotowanych posiłków.

```
SELECT
    NVL(TO_CHAR(CHI.NAME), 'ALL') AS "CHEF NAME",
    NVL(TO_CHAR(CHI.SURNAME), 'ALL') AS "CHEF SURNAME",
    NVL(TO_CHAR(MN.NAME), 'ALL') AS MENU,
    CLIENT_RATING AS "CLIENT RATING",
    "SALES COUNT"
FROM
    (
        SELECT
            CHIEF_ID,
            MENU_ID,
            NVL(TO_CHAR(INV.CLIENT_RATING), 'ALL') AS CLIENT_RATING,
            COUNT(*) AS "SALES COUNT"
        FROM
            INVOICE INV
        GROUP BY
            GROUPING SETS(
                (INV.CHIEF_ID),
                (INV.CHIEF_ID, INV.CLIENT_RATING),
                (INV.CHIEF_ID, INV.MENU_ID),
                (INV.CLIENT_RATING)
            )
    ) MAIN
    LEFT OUTER JOIN MENU MN ON MAIN.MENU_ID = MN.ID
    LEFT OUTER JOIN CHIEF CHI ON CHI.ID = MAIN.CHIEF_ID;
```

	CHEF NAME	CHEF SURNAME	MENU	CLIENT RATING	SALES COUNT
1	Adrianna	Koczara	Kids	ALL	791
2	Adrianna	Koczara	Normalne	ALL	779
3	Adrianna	Koczara	Premium	ALL	721
4	Adrianna	Koczara	Vegan	ALL	730
5	Adrianna	Koczara	ALL	ALL	3021
6	MiÅ‚osz	Radziewicz	Kids	ALL	804
7	MiÅ‚osz	Radziewicz	Normalne	ALL	744
8	MiÅ‚osz	Radziewicz	Premium	ALL	717
9	MiÅ‚osz	Radziewicz	Vegan	ALL	726
10	MiÅ‚osz	Radziewicz	ALL	ALL	2991
11	Dagmara	Kaczka	Kids	ALL	771
12	Dagmara	Kaczka	Normalne	ALL	745
13	Dagmara	Kaczka	Premium	ALL	736
14	Dagmara	Kaczka	Vegan	ALL	726
15	Dagmara	Kaczka	ALL	ALL	2978
16	Dawid	Bogaczyk	Kids	ALL	778
17	Dawid	Bogaczyk	Normalne	ALL	752

2. Zapytanie pokazuje sprzedaż z podziałem na dni tygodnia i godziny.

```
SELECT
    NVL(TO_CHAR(BIS.NAME), 'ALL') AS BISTRO,
    HOUR,
    WEEKEDAY,
    SALES
FROM
    (
        SELECT
            BISTRO_ID,
            NVL(TO_CHAR(INV.HOUR), 'ALL') AS "HOUR",
            NVL(TO_CHAR(INV.WEEKDAY), 'ALL') AS "WEEKEDAY",
            count(*) sales
        FROM
            INVOICE INV
        GROUP BY
            GROUPING SETS(
                (INV.HOUR),
                (INV.HOUR, INV.WEEKDAY),
                (INV.BISTRO_ID, INV.HOUR, INV.WEEKDAY),
                (INV.HOUR, INV.WEEKDAY)
            )
        ORDER BY
            INV.BISTRO_ID ASC,
            INV.HOUR ASC,
            INV.WEEKDAY ASC
    ) MAIN
    LEFT OUTER JOIN BISTRO BIS ON BIS.ID = MAIN.BISTRO_ID;
```

	BISTRO	HOUR	WEEKEDAY	SALES
1	NaleÅ›niczek	0	1	206
2	NaleÅ›niczek	0	2	189
3	NaleÅ›niczek	0	3	189
4	NaleÅ›niczek	0	4	174
5	NaleÅ›niczek	0	5	191
6	NaleÅ›niczek	0	6	177
7	NaleÅ›niczek	0	7	189
8	NaleÅ›niczek	1	1	188
9	NaleÅ›niczek	1	2	184
10	NaleÅ›niczek	1	3	190
11	NaleÅ›niczek	1	4	138
12	NaleÅ›niczek	1	5	161
13	NaleÅ›niczek	1	6	195
14	NaleÅ›niczek	1	7	162
15	NaleÅ›niczek	2	1	176
16	NaleÅ›niczek	2	2	162
17	NaleÅ›niczek	2	3	188

3. Zapytanie pokazuje jaki szef przygotował jaką ilość jedzenia, dodatkowo pokazana została średnia ocena klienta.

```
SELECT
    NVL(TO_CHAR(BIS.NAME), 'ALL') AS BISTRO,
    NVL(TO_CHAR(CHI.NAME), 'ALL') AS "CHEF NAME",
    NVL(TO_CHAR(CHI.SURNAME), 'ALL') AS "CHEF SURNAME",
    CLIENT_RATING AS "CLIENT RATING",
    AVERAGE_QTY AS "AVERAGE QUANTITY"
FROM
    (
        SELECT
            BISTRO_ID,
            CHIEF_ID,
            NVL(TO_CHAR(INV.CLIENT_RATING), 'ALL') AS CLIENT_RATING,
            ROUND(AVG(INV.QUANTITY), 2) as AVERAGE_QTY
        FROM
            INVOICE INV
        GROUP BY
            GROUPING SETS(
                (INV.BISTRO_ID),
                (INV.CHIEF_ID),
                (INV.CLIENT_RATING),
                (INV.BISTRO_ID, INV.CLIENT_RATING)
            )
        ORDER BY
            INV.BISTRO_ID ASC,
            INV.CHIEF_ID ASC,
            INV.CLIENT_RATING ASC
    ) MAIN
    LEFT OUTER JOIN BISTRO BIS ON BIS.ID = MAIN.BISTRO_ID
    LEFT OUTER JOIN CHIEF CHI ON CHI.ID = MAIN.CHIEF_ID;
```

	BISTRO	CHEF NAME	CHEF SURNAME	CLIENT RATING	AVERAGE QUANTITY
1	NaleÅniczek	ALL	ALL	1	2,96
2	NaleÅniczek	ALL	ALL	2	2,98
3	NaleÅniczek	ALL	ALL	3	3,01
4	NaleÅniczek	ALL	ALL	4	2,99
5	NaleÅniczek	ALL	ALL	5	2,98
6	NaleÅniczek	ALL	ALL	6	3
7	NaleÅniczek	ALL	ALL	7	3,01
8	NaleÅniczek	ALL	ALL	8	3
9	NaleÅniczek	ALL	ALL	9	3,02
10	NaleÅniczek	ALL	ALL	10	3,01
11	NaleÅniczek	ALL	ALL	ALL	3
12	Bistro MaseÅ,ko	ALL	ALL	1	3,03
13	Bistro MaseÅ,ko	ALL	ALL	2	2,96
14	Bistro MaseÅ,ko	ALL	ALL	3	2,99
15	Bistro MaseÅ,ko	ALL	ALL	4	2,98
16	Bistro MaseÅ,ko	ALL	ALL	5	3,01
17	Bistro MaseÅ,ko	ALL	ALL	6	3,01

Partition:

1. Zapytanie pokazuje % utargu danej restauracji.

```
SELECT
    BIS.NAME AS BISTRO,
    TAKINGS AS "TAKINGS  % IN YEAR"
FROM
    (
        SELECT
            DISTINCT INV.BISTRO_ID,
            INV.YEAR,
            ROUND(
                (
                    100 * sum(PRICE_VALUE) OVER (partition BY INV.BISTRO_ID, INV.YEAR) / SUM(INV.PRICE_VALUE) OVER (partition by
                     INV.YEAR)
                ),
                2
            ) TAKINGS
        FROM
            INVOICE INV
        ORDER BY
            INV.YEAR ASC
    ) MAIN
    JOIN BISTRO BIS ON BIS.ID = MAIN.BISTRO_ID;
```

	BISTRO	TAKINGS % IN YEAR
1	NaleÅ›niczek	10,39
2	NaleÅ›niczek	9,94
3	NaleÅ›niczek	9,77
4	NaleÅ›niczek	9,89
5	NaleÅ›niczek	9,9
6	NaleÅ›niczek	10,24
7	NaleÅ›niczek	9,88
8	NaleÅ›niczek	9,92
9	NaleÅ›niczek	9,77
10	NaleÅ›niczek	10,31
11	NaleÅ›niczek	9,66
12	NaleÅ›niczek	10,05
13	NaleÅ›niczek	9,41
14	NaleÅ›niczek	9,95
15	NaleÅ›niczek	10,79
16	NaleÅ›niczek	9,8
17	NaleÅ›niczek	10,06

2. Zapytanie pokazuje najczęściej kupowany produkt w menu

```
SELECT
    PROD.NAME AS PRODUCT,
    MN.NAME AS MENU,
    PIM AS "% PRODUCT IN MENU"
FROM
    (
        SELECT
            DISTINCT INV.PRODUCT_ID,
            INV.MENU_ID,
            ROUND(
                (
                    100 * COUNT(INV.PRODUCT_ID) OVER (partition BY INV.PRODUCT_ID, INV.MENU_ID) / COUNT(INV.PRODUCT_ID) OVER (
                    partition by INV.MENU_ID)
                ),
                2
            ) as PIM
        FROM
            INVOICE INV
        ORDER BY
            INV.MENU_ID
    ) MAIN
    JOIN PRODUCT PROD ON PROD.ID = MAIN.PRODUCT_ID
    JOIN MENU MN ON MN.ID = MAIN.MENU_ID;
```

	PRODUCT	MENU	% PRODUCT IN MENU
1	Tea	Kids	4,53
2	Tea	Normalne	4,57
3	Tea	Premium	4,59
4	Tea	Vegan	4,56
5	Water	Kids	4,39
6	Water	Normalne	4,58
7	Water	Premium	4,49
8	Water	Vegan	4,49
9	Mirinda	Kids	4,59
10	Mirinda	Normalne	4,54
11	Mirinda	Premium	4,48
12	Mirinda	Vegan	4,5
13	Coffie	Kids	4,66
14	Coffie	Normalne	4,51
15	Coffie	Premium	4,65
16	Coffie	Vegan	4,58
17	Grass salad	Kids	4,54

3. Zapytanie pokazuje rabaty w danym bistro.

```
SELECT
    BIS.NAME AS BISTRO,
    BN.NAME AS BONUS,
    BIB AS "% BONUS IN BISTRO"
FROM
    (
        SELECT
            DISTINCT INV.BONUS_ID,
            INV.BISTRO_ID,
            ROUND(
                (
                    100 * COUNT(INV.BONUS_ID) OVER (partition BY INV.BONUS_ID, INV.BISTRO_ID) / COUNT(INV.BONUS_ID) OVER (
                    partition by INV.BISTRO_ID)
                ),
                2
            ) as BIB
        FROM
            INVOICE INV
        ORDER BY
            INV.BISTRO_ID
    ) MAIN
    JOIN BISTRO BIS ON MAIN.BISTRO_ID = BIS.ID
    JOIN BONUS BN ON BN.ID = MAIN.BONUS_ID;
```

	BISTRO	BONUS	% BONUS IN BISTRO
1	JabÅ,ko	Brak	20,3
2	Bistro MaseÅ,ko	Brak	20,21
3	Tanio i smacznie	Brak	19,91
4	Smaczne Bistro	Brak	19,95
5	NaleÅ›niczek	Brak	20,37
6	KrÃ³wka	Brak	19,76
7	Melon i SpÃ³Å,ka	Brak	20,29
8	Burger & Co	Brak	20,27
9	Bistro BuÅ,eczka	Brak	19,86
10	Gruszka	Brak	20,23
11	Burger & Co	Dla_studentÃ³w	19,86
12	Tanio i smacznie	Dla_studentÃ³w	20,1
13	Bistro MaseÅ,ko	Dla_studentÃ³w	19,96
14	Bistro BuÅ,eczka	Dla_studentÃ³w	19,86
15	Smaczne Bistro	Dla_studentÃ³w	19,94
16	NaleÅ›niczek	Dla_studentÃ³w	20,12
17	Melon i SpÃ³Å,ka	Dla_studentÃ³w	19,83

Rank:

1. Ranking restauracji ze względu na sprzedaż

```
SELECT
    BIS.ID,
    BIS.NAME,
    MAIN.COUNT "SALES COUNT",
    DENSE_RANK() OVER (
        ORDER BY
            MAIN.COUNT DESC
    ) AS RANK
FROM
    (
        SELECT
            INV.BISTRO_ID as ID,
            COUNT(*) AS COUNT
        FROM
            INVOICE INV
        GROUP BY
            INV.BISTRO_ID
    ) MAIN
    JOIN BISTRO BIS ON BIS.ID = MAIN.ID;
```

	ID	NAME	SALES COUNT	RANK
1	9	Gruszka	30211	1
2	1	NaleÅ›niczek	30126	2
3	8	JabÅ‚ko	30085	3
4	4	KrÃ³wka	30065	4
5	5	Smaczne Bistro	30003	5
6	6	Tanio i smacznie	30002	6
7	7	Bistro BuÅ‚eczka	30001	7
8	10	Melon i SpÃ³Å‚ka	29912	8
9	3	Burger & Co	29833	9
10	2	Bistro MaseÅ‚ko	29762	10

2. Ranking ze względu na oceny klientów

```
SELECT
    BIS.ID,
    BIS.NAME,
    MAIN.AVG_RATING AVG_RATING,
    DENSE_RANK() OVER (
        ORDER BY
            MAIN.AVG_RATING DESC
    ) AS RANK
FROM
    (
        SELECT
            INV.BISTRO_ID as ID,
            ROUND(AVG(INV.CLIENT_RATING), 2) AVG_RATING
        FROM
            INVOICE INV
        GROUP BY
            INV.BISTRO_ID
    ) MAIN
    JOIN BISTRO BIS ON BIS.ID = MAIN.ID;
```

	ID	NAME	AVG_RATING	RANK
1	4	KrÃ³wka	5,53	1
2	2	Bistro MaseÅ‚ko	5,53	1
3	8	JabÅ‚ko	5,53	1
4	1	NaleÅ›niczek	5,52	2
5	9	Gruszka	5,51	3
6	10	Melon i SpÃ³Å‚ka	5,51	3
7	7	Bistro BuÅ‚eczka	5,5	4
8	5	Smaczne Bistro	5,49	5
9	3	Burger & Co	5,48	6
10	6	Tanio i smacznie	5,48	6

3. Ranking szefów kuchni

```sql
SELECT
    CHI.ID,
    CHI.NAME,
    CHI.SURNAME,
    MAIN.AVG_RATING,
    DENSE_RANK() OVER (
        ORDER BY
            MAIN.AVG_RATING DESC
    ) AS RANK
FROM
    (
        SELECT
            INV.CHIEF_ID AS ID,
            ROUND(AVG(INV.CLIENT_RATING), 2) AVG_RATING
        FROM
            INVOICE INV
        GROUP BY
            INV.CHIEF_ID
    ) MAIN
    JOIN CHIEF CHI ON CHI.ID = MAIN.ID
```

	ID	NAME	SURNAME	AVG_RATING	RANK
1	41	Kornelia	Ä†wirko	5,64	1
2	38	Wiktor	Gajowiak	5,63	2
3	7	PrzemysÅ,aw	Dolega	5,62	3
4	100	Oskar	Å□yp	5,61	4
5	85	Wojciech	Ropiak	5,61	4
6	35	Marcelina	Macias	5,61	4
7	63	PaweÅ,	Grzelec	5,61	4
8	21	Kaja	Szyc	5,59	5
9	17	Nicole	Uroda	5,59	5
10	54	Adam	Chatys	5,58	6
11	4	Dawid	Bogaczyk	5,58	6
12	93	Bruno	PachoÅ,ek	5,58	6
13	19	Aniela	Wasiuk	5,57	7
14	95	Hubert	Steuer	5,56	8
15	62	Marek	Siarkiewicz	5,56	8
16	37	MiÅ,osz	Jajko	5,56	8
17	9	RafaÅ,	Kucharz	5,56	8

Rollup:

1. Zapytanie pokazuje utarg w danych miesiącach i latach.

```
SELECT
    nvl(bis.name, 'ALL') bistro_name,
    inv.client_rating,
    COUNT(inv.id)        invoice_count,
    SUM(inv.price_value) takings
FROM
        invoice inv
    JOIN bistro bis ON bis.id = inv.bistro_id
GROUP BY
    rollup(bis.name, inv.client_rating)
ORDER BY
    bis.name;
```

	BISTRO_NAME	CLIENT_RATING	INVOICE_COUNT	TAKINGS
1	Bistro BuÅ‚eczka	6	2944	1455307
2	Bistro BuÅ‚eczka	7	2992	1484551
3	Bistro BuÅ‚eczka	8	2963	1483727
4	Bistro BuÅ‚eczka	9	2992	1507750
5	Bistro BuÅ‚eczka	10	3081	1544382
6	Bistro BuÅ‚eczka	(null)	30001	15016841
7	Bistro BuÅ‚eczka	4	2977	1484444
8	Bistro BuÅ‚eczka	3	2995	1503159
9	Bistro BuÅ‚eczka	2	3049	1520027
10	Bistro BuÅ‚eczka	1	2997	1513848
11	Bistro BuÅ‚eczka	5	3011	1519646
12	Bistro MaseÅ‚ko	5	2974	1495707
13	Bistro MaseÅ‚ko	4	3016	1495653
14	Bistro MaseÅ‚ko	3	2996	1509477
15	Bistro MaseÅ‚ko	2	2913	1448826
16	Bistro MaseÅ‚ko	1	2868	1419869
17	Bistro MaseÅ‚ko	6	3004	1496493

2. Zapytanie zwraca liczbę sprzedaży w danym bistro w danym miesiącu i roku danego produktu.

```
SELECT
    nvl(bis.name, 'OVERALL RATING') bistro_name,
    inv.month,
    ROUND(AVG(INV.client_rating),6) avg_rate
FROM
        invoice inv
JOIN bistro bis ON bis.id = inv.bistro_id
GROUP BY
    rollup(bis.name, inv.month)
ORDER BY
    bis.name, inv.month;
```

	BISTRO_NAME	MONTH	AVG_RATE
1	Bistro BuÅ,eczka	1	5,509031
2	Bistro BuÅ,eczka	2	5,361451
3	Bistro BuÅ,eczka	3	5,430237
4	Bistro BuÅ,eczka	4	5,512393
5	Bistro BuÅ,eczka	5	5,657792
6	Bistro BuÅ,eczka	6	5,572005
7	Bistro BuÅ,eczka	7	5,470754
8	Bistro BuÅ,eczka	8	5,701747
9	Bistro BuÅ,eczka	9	5,456513
10	Bistro BuÅ,eczka	10	5,434194
11	Bistro BuÅ,eczka	11	5,501386
12	Bistro BuÅ,eczka	12	5,428103
13	Bistro BuÅ,eczka	(null)	5,502917
14	Bistro MaseÅ,ko	1	5,549324
15	Bistro MaseÅ,ko	2	5,622015
16	Bistro MaseÅ,ko	3	5,47621
17	Bistro MaseÅ,ko	4	5,557959

3. Zapytanie zwraca liczbę zamówień danego menu w restauracji

```
SELECT
    nvl(bis.name, 'OVERALL') bistro_name,
    nvl(poc.type, 'OVERALL TYPE') TYPE,
    count(poc.type)
FROM
        invoice inv
JOIN bistro bis ON bis.id = inv.bistro_id
JOIN place_of_consumption poc ON poc.id = inv.poc_id
GROUP BY
    rollup(bis.name, poc.type)
ORDER BY
    bis.name, poc.type
;
```

	BISTRO_NAME	TYPE	COUNT(POC.TYPE)
1	Bistro BuÅ‚eczka	Dostawa	10021
2	Bistro BuÅ‚eczka	Na_miejscu	9819
3	Bistro BuÅ‚eczka	Na_wynos	10161
4	Bistro BuÅ‚eczka	OVERALL TYPE	30001
5	Bistro MaseÅ‚ko	Dostawa	9781
6	Bistro MaseÅ‚ko	Na_miejscu	9933
7	Bistro MaseÅ‚ko	Na_wynos	10048
8	Bistro MaseÅ‚ko	OVERALL TYPE	29762
9	Burger & Co	Dostawa	9965
10	Burger & Co	Na_miejscu	9869
11	Burger & Co	Na_wynos	9999
12	Burger & Co	OVERALL TYPE	29833
13	Gruszka	Dostawa	9998
14	Gruszka	Na_miejscu	9945
15	Gruszka	Na_wynos	10268
16	Gruszka	OVERALL TYPE	30211
17	JabÅ‚ko	Dostawa	9970

Window:

1. Zapytanie zwraca przyrosty dochodu z miesiąca na miesiąc

```
SELECT
    YEAR,
    MONTH,
    PV AS "THIS MONTH TAKINGS",
    Sum(PV) over (
        PARTITION BY YEAR
        ORDER BY
            MONTH RANGE BETWEEN unbounded preceding
            AND CURRENT ROW
    ) AS "TAKINGS SUM TO THIS MONTH"
from
    (
        SELECT
            MONTH,
            YEAR,
            SUM(PRICE_VALUE) PV
        FROM
            INVOICE
        GROUP BY
            MONTH,
            YEAR
    )
ORDER BY
    YEAR,
    MONTH;
```

	YEAR	MONTH	THIS MONTH TAKINGS	TAKINGS SUM TO THIS MONTH
1	1997	1	484083	484083
2	1997	2	461167	945250
3	1997	3	453292	1398542
4	1997	4	440684	1839226
5	1997	5	449444	2288670
6	1997	6	460190	2748860
7	1997	7	481892	3230752
8	1997	8	444516	3675268
9	1997	9	491739	4167007
10	1997	10	437610	4604617
11	1997	11	480326	5084943
12	1997	12	444104	5529047
13	1998	1	442104	442104
14	1998	2	441319	883423
15	1998	3	476546	1359969
16	1998	4	489257	1849226
17	1998	5	481929	2331155

2. Zapytanie zwraca różnicę w ilości sprzedaży między bieżącym miesiącem a poprzednim.

```sql
SELECT
    YEAR,
    MONTH,
    L_SPRZED AS INVOICES_COUNT,
    NVL(
        L_SPRZED -(
            Lag(L_SPRZED, 1) over (
                ORDER BY
                    YEAR,
                    MONTH
            )
        ),
        0
    ) AS "DIFF BETWEEN MONTHS"
from
    (
        SELECT
            MONTH,
            YEAR,
            COUNT(*) L_SPRZED
        FROM
            INVOICE
        GROUP BY
            MONTH,
            YEAR
    )
ORDER BY
    YEAR ASC,
    MONTH ASC;
```

	YEAR	MONTH	INVOICES_COUNT	DIFF BETWEEN MONTHS
1	1997	1	992	0
2	1997	2	938	-54
3	1997	3	918	-20
4	1997	4	918	0
5	1997	5	895	-23
6	1997	6	912	17
7	1997	7	965	53
8	1997	8	916	-49
9	1997	9	951	35
10	1997	10	898	-53
11	1997	11	952	54
12	1997	12	921	-31
13	1998	1	909	-12
14	1998	2	884	-25
15	1998	3	937	53
16	1998	4	978	41
17	1998	5	939	-39

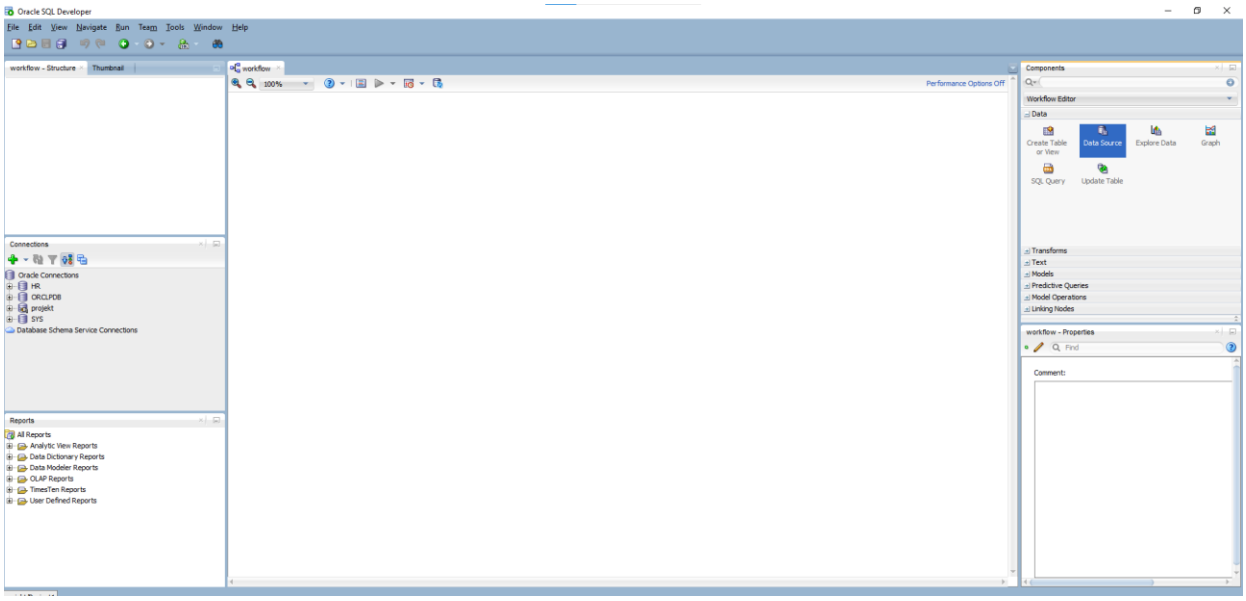
3. Zapytanie zwraca, jak się zmieniała średnia ocen klientów.

```
SELECT
    YEAR,
    MONTH,
    A_RAT AS AVG_RATING,
    NVL(
        A_RAT -(
            Lag(A_RAT, 1) over (
                ORDER BY
                    YEAR,
                    MONTH
            )
        ),
        0
    ) AS "DIFF BETWEEN MONTHS"
from
    (
        SELECT
            MONTH,
            YEAR,
            ROUND(AVG(CLIENT_RATING), 2) A_RAT
        FROM
            INVOICE
        GROUP BY
            MONTH,
            YEAR
    )
ORDER BY
    YEAR ASC,
    MONTH ASC;
```

	YEAR	MONTH	AVG_RATING	DIFF BETWEEN MONTHS
1	1997	1	5,42	0
2	1997	2	5,34	-0,08
3	1997	3	5,48	0,14
4	1997	4	5,57	0,09
5	1997	5	5,38	-0,19
6	1997	6	5,31	-0,07
7	1997	7	5,57	0,26
8	1997	8	5,47	-0,1
9	1997	9	5,46	-0,01
10	1997	10	5,54	0,08
11	1997	11	5,54	0
12	1997	12	5,47	-0,07
13	1998	1	5,32	-0,15
14	1998	2	5,58	0,26
15	1998	3	5,57	-0,01
16	1998	4	5,65	0,08
17	1998	5	5,46	-0,19

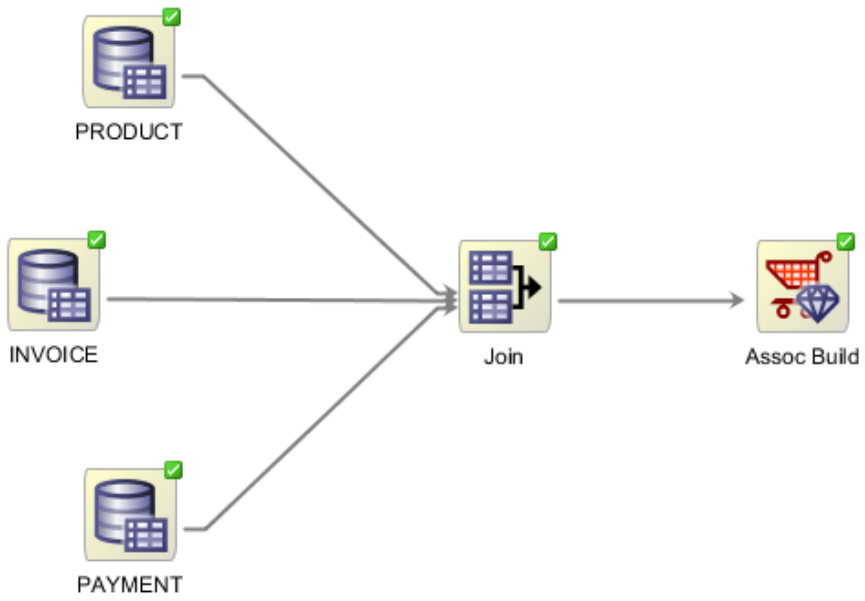
6. Eksploracja danych

Po uruchomieniu data minera należy stworzyć nowy projekt. Dla każdego podejścia eksploracji danych został stworzony osobny workflow. Do zbudowania schematu zostało wykorzystane menu po prawej stronie, z którego możemy tworzyć schematy.

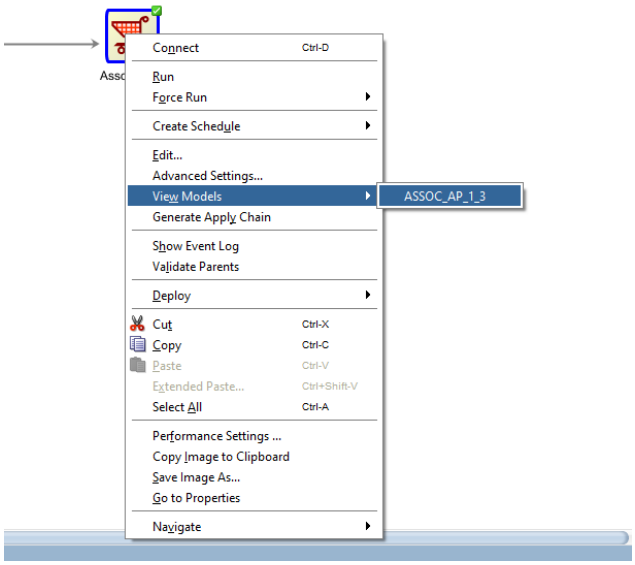


1. Reguły asocjacyjne - często wybierane produkty w restauracji.

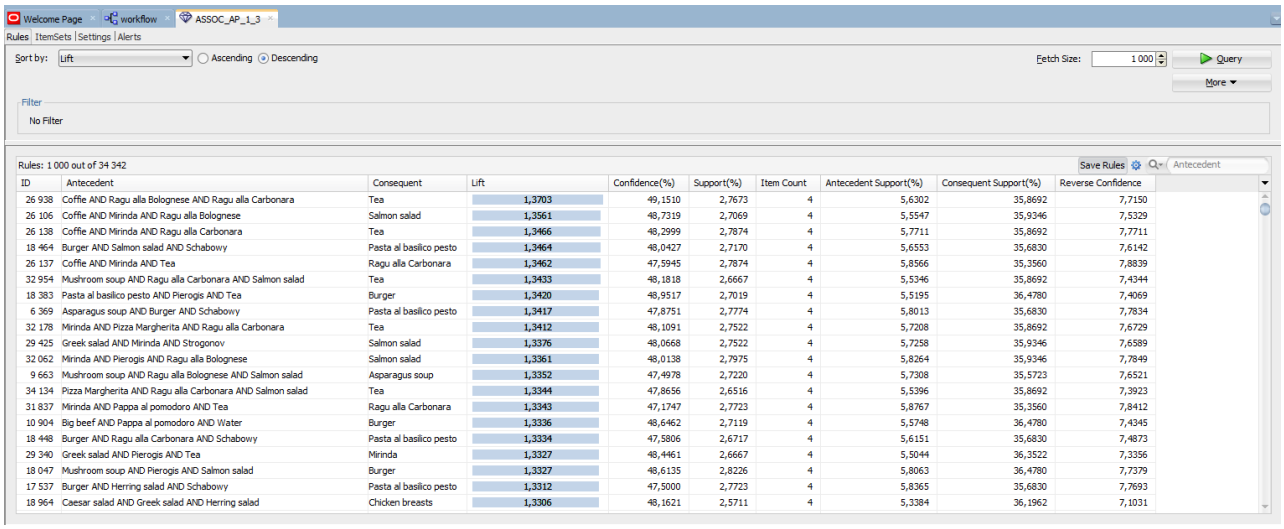
Schemat:



Po uruchomieniu modelu możemy podejrzeć wynik. Jest on dostępny po kliknięciu prawym klawiszem na ostatni moduł.



Otrzymany wynik:


Welcome Page | workflow | ASSOC_AP_1_3

Rules | ItemSets | Settings | Alerts

Sort by: Lift | Ascending | Descending | Fetch Size: 1 000 | Query | More

Filter: No Filter

Rules: 1 000 out of 34 342 | Save Rules | Antecedent

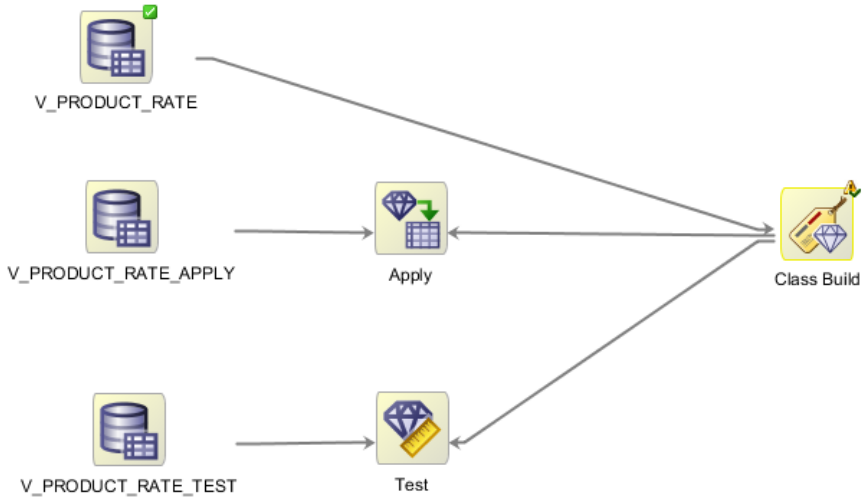
ID	Antecedent	Consequent	Lift	Confidence(%)	Support(%)	Item Count	Antecedent Support(%)	Consequent Support(%)	Reverse Confidence
26 938	Coffie AND Ragu alla Bolognese AND Ragu alla Carbonara	Tea	1,3703	49,1510	2,7673	4	5,6302	35,8692	7,7150
26 106	Coffie AND Mirinda AND Ragu alla Bolognese	Salmon salad	1,3561	48,7319	2,7069	4	5,5547	35,9346	7,5329
26 138	Coffie AND Mirinda AND Ragu alla Carbonara	Tea	1,3466	48,2999	2,7874	4	5,7711	35,8692	7,7711
18 464	Burger AND Salmon salad AND Schabowy	Pasta al basilico pesto	1,3464	48,0427	2,7170	4	5,6553	35,6830	7,6142
26 137	Coffie AND Mirinda AND Tea	Ragu alla Carbonara	1,3462	47,5945	2,7874	4	5,8566	35,3560	7,8839
32 954	Mushroom soup AND Ragu alla Carbonara AND Salmon salad	Tea	1,3433	48,1818	2,6667	4	5,5346	35,8692	7,4344
18 383	Pasta al basilico pesto AND Pierogis AND Tea	Burger	1,3420	48,9517	2,7019	4	5,5195	36,4780	7,4069
6 369	Asparagus soup AND Burger AND Schabowy	Pasta al basilico pesto	1,3417	47,8751	2,7774	4	5,8013	35,6830	7,7834
32 178	Mirinda AND Pizza Margherita AND Ragu alla Carbonara	Tea	1,3412	48,1091	2,7522	4	5,7208	35,8692	7,6729
29 425	Greek salad AND Mirinda AND Strogonov	Salmon salad	1,3376	48,0668	2,7522	4	5,7258	35,9346	7,6589
32 062	Mirinda AND Pierogis AND Ragu alla Bolognese	Salmon salad	1,3361	48,0138	2,7975	4	5,8264	35,9346	7,7849
9 663	Mushroom soup AND Ragu alla Bolognese AND Salmon salad	Asparagus soup	1,3352	47,4978	2,7220	4	5,7308	35,5723	7,6521
34 134	Pizza Margherita AND Ragu alla Carbonara AND Salmon salad	Tea	1,3344	47,8656	2,6516	4	5,5396	35,8692	7,3923
31 837	Mirinda AND Pappa al pomodoro AND Tea	Ragu alla Carbonara	1,3343	47,1747	2,7723	4	5,8767	35,3560	7,8412
10 904	Big beef AND Pappa al pomodoro AND Water	Burger	1,3336	48,6462	2,7119	4	5,5748	36,4780	7,4345
18 448	Burger AND Ragu alla Carbonara AND Schabowy	Pasta al basilico pesto	1,3334	47,5806	2,6717	4	5,6151	35,6830	7,4873
29 340	Greek salad AND Pierogis AND Tea	Mirinda	1,3327	48,4461	2,6667	4	5,5044	36,3522	7,3356
18 047	Mushroom soup AND Pierogis AND Salmon salad	Burger	1,3327	48,6135	2,8226	4	5,8063	36,4780	7,7379
17 537	Burger AND Herring salad AND Schabowy	Pasta al basilico pesto	1,3312	47,5000	2,7723	4	5,8365	35,6830	7,7693
18 964	Caesar salad AND Greek salad AND Herring salad	Chicken breasts	1,3306	48,1621	2,5711	4	5,3384	36,1962	7,1031

Zbudowany model pozwolił nam dowiedzieć się jakie wybory podejmują nasi klienci podczas zamawiania potraw w naszych restauracjach. Widzimy, że najczęściej biorą oni kawę oraz różne makarony lub Mirinda. W przypadku restauracji pozwoli to nam na określenie jakie promocje możemy zaoferować klientom, podnosząc sprzedaż. Pokazuje to też szefom kuchni jakie produkty powinny być rozwijane w karcie dań, aby klienci byli zadowoleni.

2. Klasyfikacja

Przygotowany problem pozwoli ocenić nam jakie potrawy interesują naszych klientów i co najlepiej rozwijać.

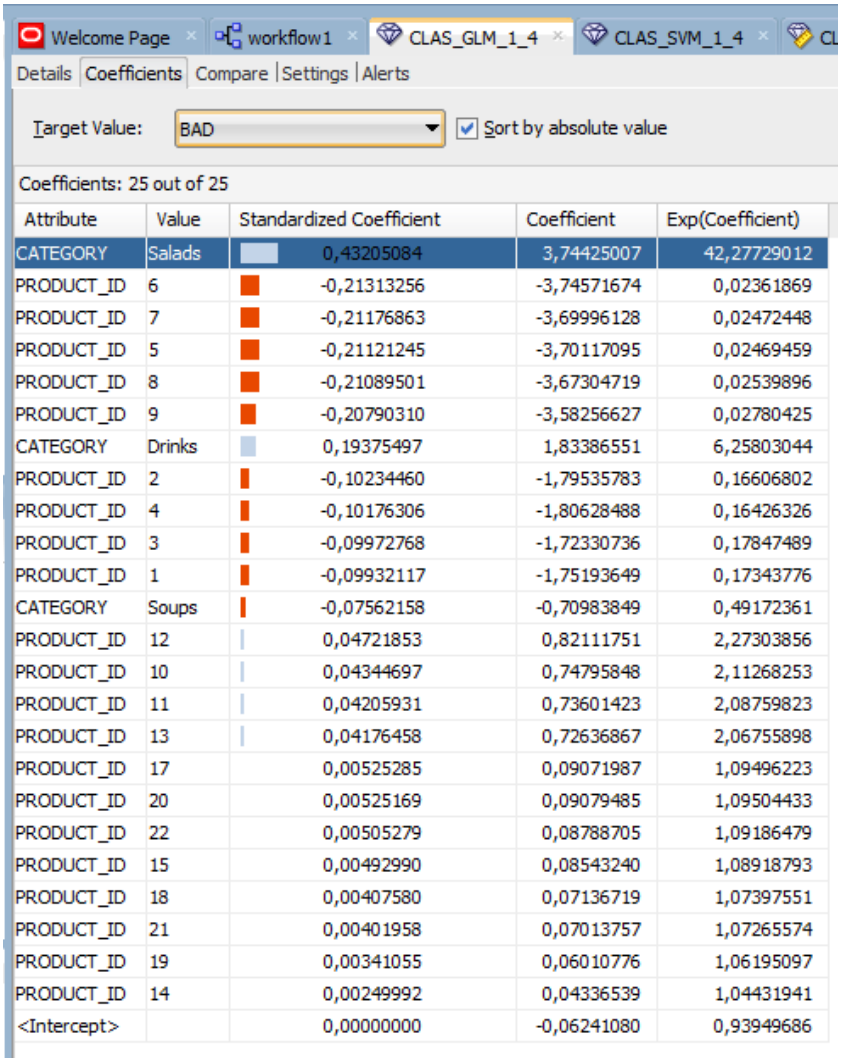
Schemat został zbudowany w taki sam sposób jak poprzedni, ale z wykorzystaniem innych bloczków.



Aby można było podzielić dane na testowe, uczące oraz weryfikujące zostały stworzone 3 widoki dzielące je na równe części.

```sql
Create or replace view V_PRODUCT_RATE AS
SELECT
    inv.product_id,
    pr.category,
    inv.id,
    CASE
            WHEN inv.client_rating < 6  THEN 'GOOD' ELSE 'BAD'
    END rating
FROM invoice inv
JOIN product pr ON inv.product_id = pr.id
                and inv.id < 66000;


Create or replace view V_PRODUCT_RATE_TEST AS
SELECT
    inv.product_id,
    pr.category,
    inv.id,
    CASE
           WHEN inv.client_rating < 6  THEN 'GOOD' ELSE 'BAD'
    END rating
FROM invoice inv
JOIN product pr ON inv.product_id = pr.id
                and inv.id between 66000 and 132000;

Create or replace view V_PRODUCT_RATE_APPLY AS
SELECT
    inv.product_id,
    pr.category,
    inv.id,
    CASE
           WHEN inv.client_rating < 6  THEN 'GOOD' ELSE 'BAD'
    END rating
FROM invoice inv
JOIN product pr ON inv.product_id = pr.id
                and inv.id between 132000 AND 200000;
```

Po uruchomieniu otrzymujemy wyniki:



Attribute	Value	Standardized Coefficient	Coefficient	Exp(Coefficient)
CATEGORY	Salads	0,43205084	3,74425007	42,27729012
PRODUCT_ID	6	-0,21313256	-3,74571674	0,02361869
PRODUCT_ID	7	-0,21176863	-3,69996128	0,02472448
PRODUCT_ID	5	-0,21121245	-3,70117095	0,02469459
PRODUCT_ID	8	-0,21089501	-3,67304719	0,02539896
PRODUCT_ID	9	-0,20790310	-3,58256627	0,02780425
CATEGORY	Drinks	0,19375497	1,83386551	6,25803044
PRODUCT_ID	2	-0,10234460	-1,79535783	0,16606802
PRODUCT_ID	4	-0,10176306	-1,80628488	0,16426326
PRODUCT_ID	3	-0,09972768	-1,72330736	0,17847489
PRODUCT_ID	1	-0,09932117	-1,75193649	0,17343776
CATEGORY	Soups	-0,07562158	-0,70983849	0,49172361
PRODUCT_ID	12	0,04721853	0,82111751	2,27303856
PRODUCT_ID	10	0,04344697	0,74795848	2,11268253
PRODUCT_ID	11	0,04205931	0,73601423	2,08759823
PRODUCT_ID	13	0,04176458	0,72636867	2,06755898
PRODUCT_ID	17	0,00525285	0,09071987	1,09496223
PRODUCT_ID	20	0,00525169	0,09079485	1,09504433
PRODUCT_ID	22	0,00505279	0,08788705	1,09186479
PRODUCT_ID	15	0,00492990	0,08543240	1,08918793
PRODUCT_ID	18	0,00407580	0,07136719	1,07397551
PRODUCT_ID	21	0,00401958	0,07013757	1,07265574
PRODUCT_ID	19	0,00341055	0,06010776	1,06195097
PRODUCT_ID	14	0,00249992	0,04336539	1,04431941
<Intercept>		0,00000000	-0,06241080	0,93949686

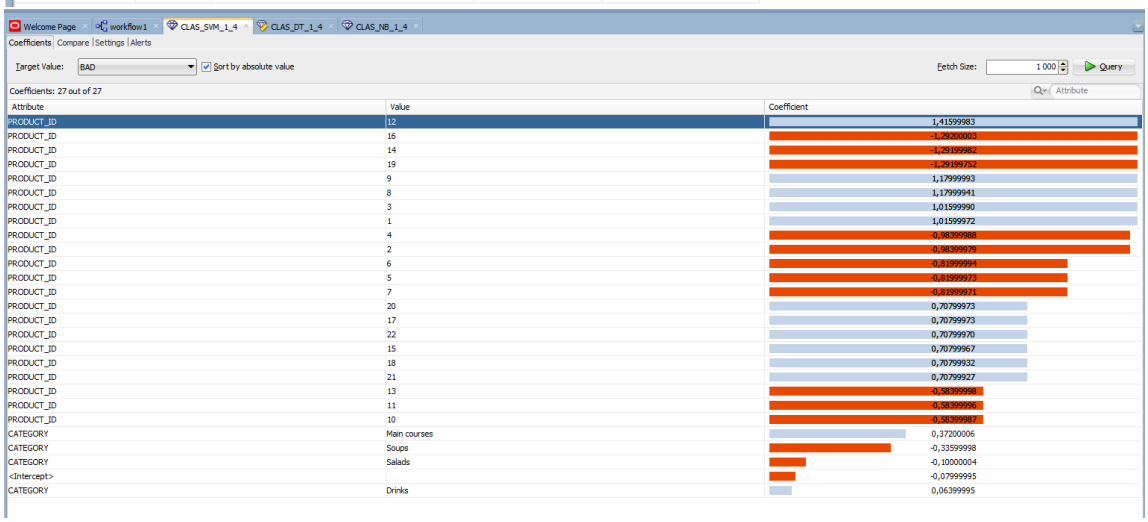
Welcome Page | workflow1 | CLAS_GLM_1_4 | CLAS_SVM_1_4 | CL

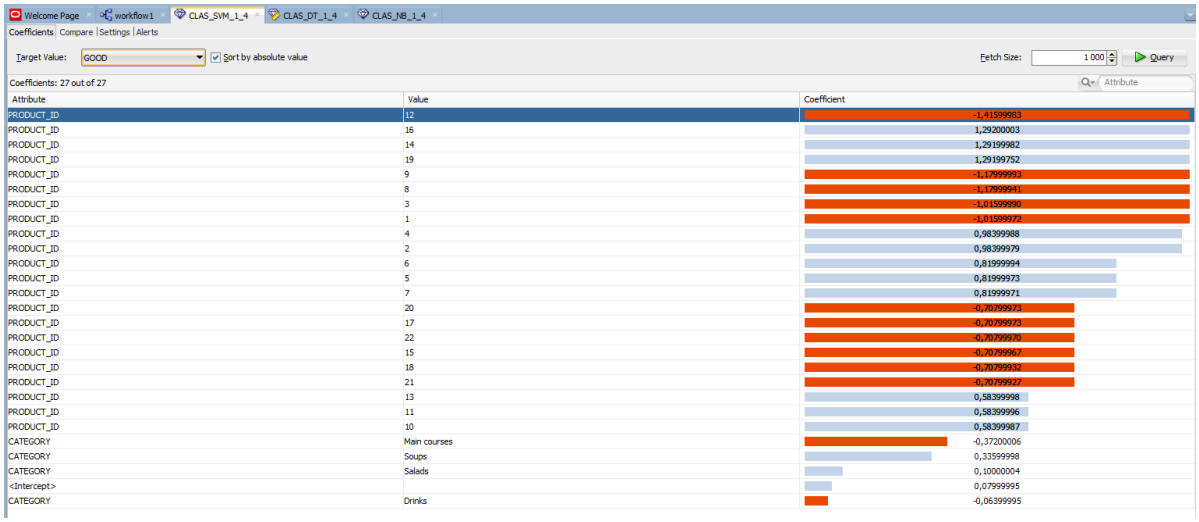
Details | Coefficients | Compare | Settings | Alerts

Target Value: GOOD Sort by absolute value

Coefficients: 25 out of 25

Attribute	Value	Standardized Coefficient	Coefficient	Exp(Coefficient)
CATEGORY	Salads	-0,43205084	-3,74425007	0,02365336
PRODUCT_ID	6	0,21313256	3,74571674	42,33934245
PRODUCT_ID	7	0,21176863	3,69996128	40,44573841
PRODUCT_ID	5	0,21121245	3,70117095	40,49469395
PRODUCT_ID	8	0,21089501	3,67304719	39,37169646
PRODUCT_ID	9	0,20790310	3,58256627	35,96572041
CATEGORY	Drinks	-0,19375497	-1,83386551	0,15979468
PRODUCT_ID	2	0,10234460	1,79535783	6,02162907
PRODUCT_ID	4	0,10176306	1,80628488	6,08778852
PRODUCT_ID	3	0,09972768	1,72330736	5,60302909
PRODUCT_ID	1	0,09932117	1,75193649	5,76575718
CATEGORY	Soups	0,07562158	0,70983849	2,03366277
PRODUCT_ID	12	-0,04721853	-0,82111751	0,43993974
PRODUCT_ID	10	-0,04344697	-0,74795848	0,47333188
PRODUCT_ID	11	-0,04205931	-0,73601423	0,47901938
PRODUCT_ID	13	-0,04176458	-0,72636867	0,48366214
PRODUCT_ID	17	-0,00525285	-0,09071987	0,91327351
PRODUCT_ID	20	-0,00525169	-0,09079485	0,91320504
PRODUCT_ID	22	-0,00505279	-0,08788705	0,91586431
PRODUCT_ID	15	-0,00492990	-0,08543240	0,91811520
PRODUCT_ID	18	-0,00407580	-0,07136719	0,93111993
PRODUCT_ID	21	-0,00401958	-0,07013757	0,93226556
PRODUCT_ID	19	-0,00341055	-0,06010776	0,94166306
PRODUCT_ID	14	-0,00249992	-0,04336539	0,95756144
<Intercept>		0,00000000	0,06241080	1,06439952





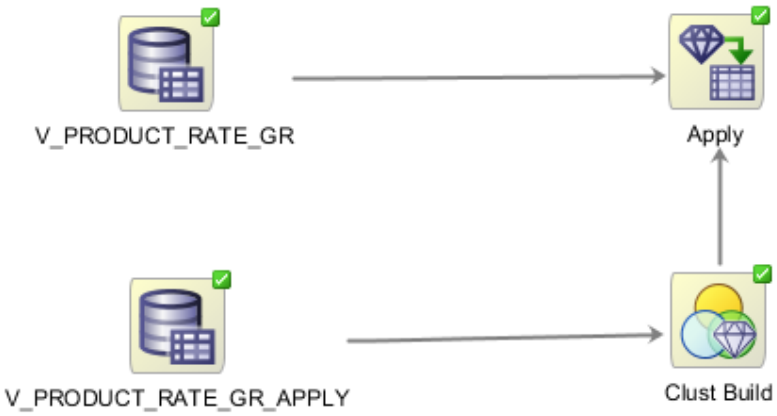
Przygotowany model pozwolił nam zobaczyć jaki produkt oraz kategoria jest najlepiej oceniana przez naszych klientów. Pozwoli to restauracji na rozwój produktów, które najlepiej smakują klientom i zmaksymalizują zysk firmy.

3. Grupowanie

W grupowaniu również poruszony został problem dań oraz ocen klientów. Chcielibyśmy się dowiedzieć jak potrawy lub grupy są oceniane. Pozwoli to nam lepiej profilować naszych klientów.

Zostało wykonane podobnie do dwóch poprzednich schematów jednak z innymi bloczkami. Aby podzielić dane na testowe oraz uczące zostały podzielone w widokach.

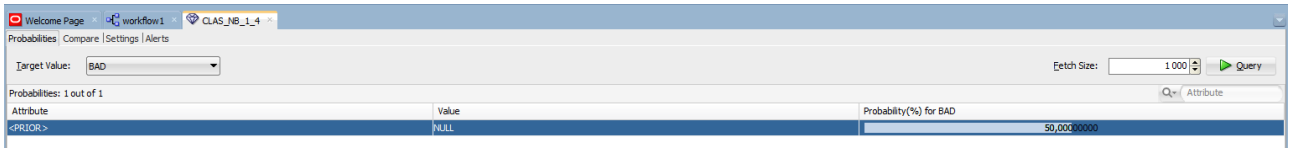
Schemat:

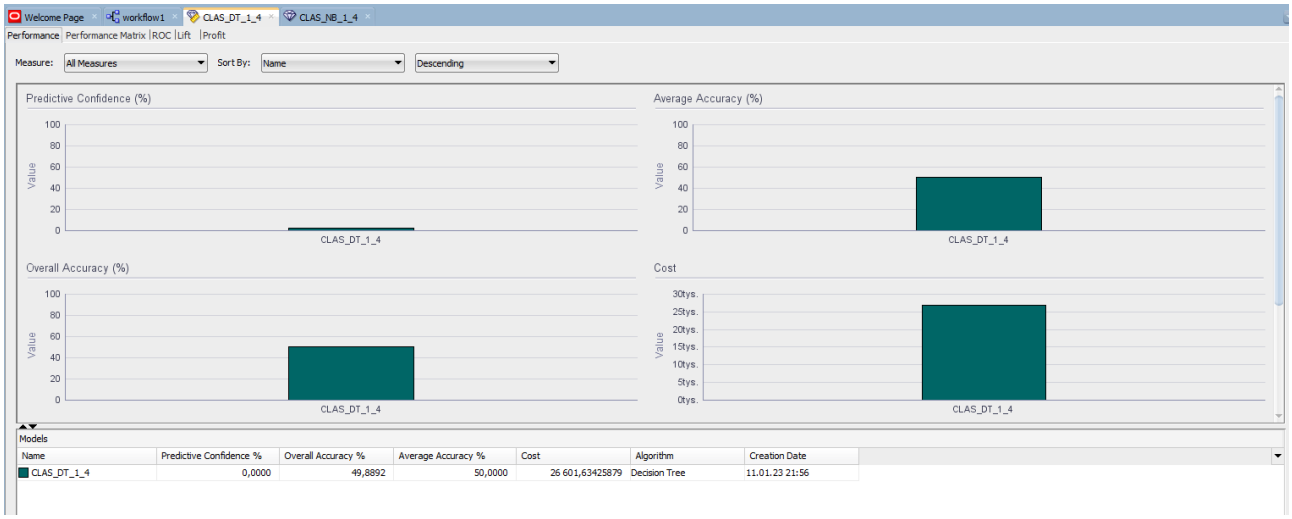


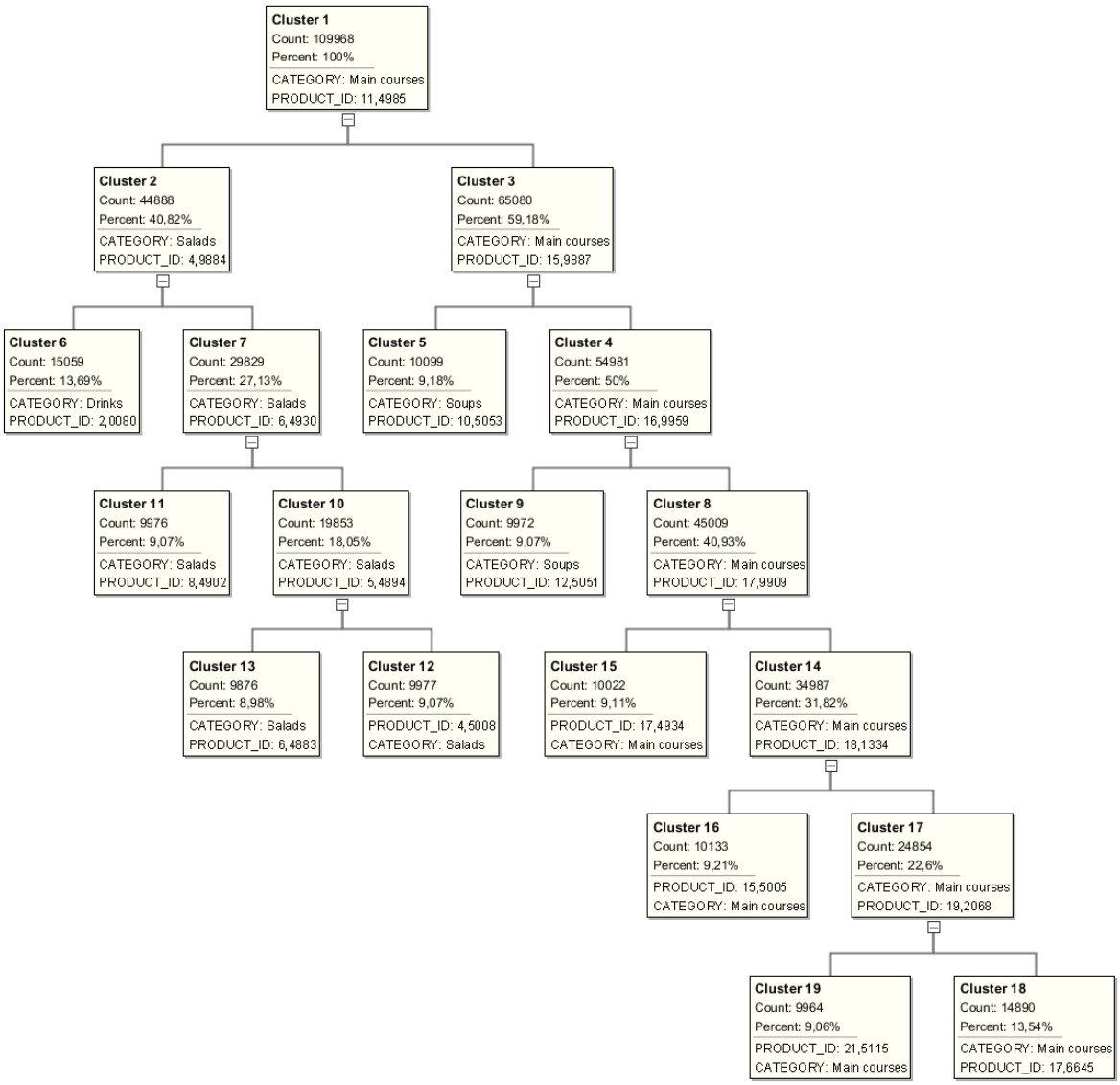
Widoki:

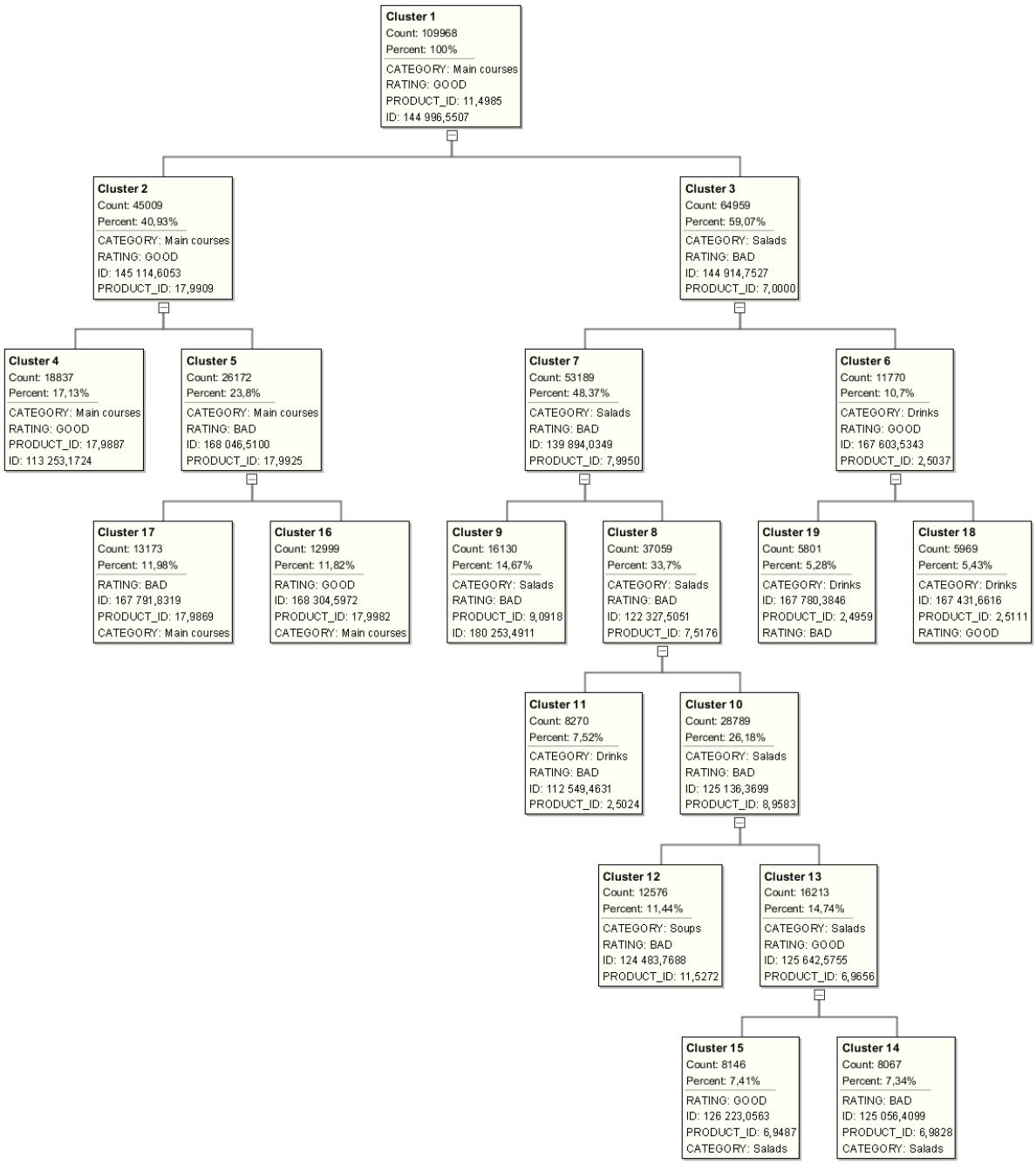
```
Create or replace view V_PRODUCT_RATE_GR AS
SELECT
    inv.product_id,
    pr.category,
    inv.id,
    CASE
            WHEN inv.client_rating < 6  THEN 'GOOD' ELSE 'BAD'
    END rating
FROM invoice inv
JOIN product pr ON inv.product_id = pr.id
                and inv.id < 90000;

Create or replace view V_PRODUCT_RATE_GR_APPLY AS
SELECT
    inv.product_id,
    pr.category,
    inv.id,
    CASE
            WHEN inv.client_rating < 6  THEN 'GOOD' ELSE 'BAD'
    END rating
FROM invoice inv
JOIN product pr ON inv.product_id = pr.id
                and inv.id >= 90000;
```

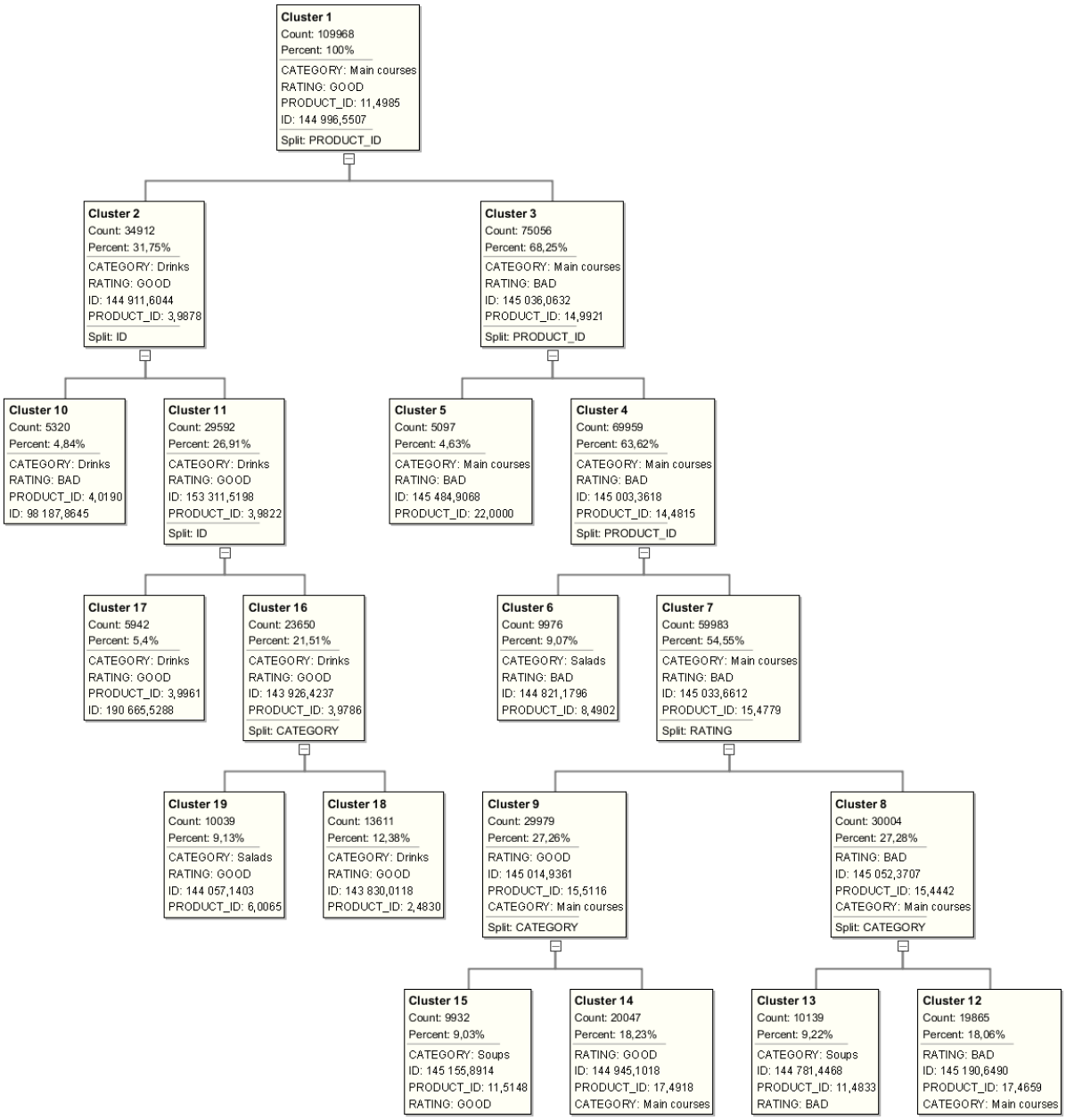
Otrzymane wyniki:


Welcome Page
workflow1
CLAS_NB_1_4
Probabilities
Compare
Settings
Alerts
Target Value:
BAD
Fetch Size:
1 000
Query
Probabilities: 1 out of 1
Attribute
Attribute
Value
Probability(%) for BAD
<PRIOR>
NULL
50,00000000


Welcome Page
workflow1
CLAS_DT_1_4
CLAS_NB_1_4
Performance
Performance Matrix
ROC
Lift
Profit
Measure:
All Measures
Sort By:
Name
Descending
Predictive Confidence (%)
Average Accuracy (%)
Overall Accuracy (%)
Cost
Value
CLAS_DT_1_4
30tys.
25tys.
20tys.
15tys.
10tys.
5tys.
0tys.
Models
Name
Predictive Confidence %
Overall Accuracy %
Average Accuracy %
Cost
Algorithm
Creation Date
CLAS_DT_1_4
0,0000
49,8892
50,0000
26 601,63425879
Decision Tree
11.01.23 21:56


Cluster 1
Count: 109968
Percent: 100%
CATEGORY: Main courses
PRODUCT_ID: 11,4985
Cluster 2
Count: 44888
Percent: 40,82%
CATEGORY: Salads
PRODUCT_ID: 4,9884
Cluster 3
Count: 65080
Percent: 59,18%
CATEGORY: Main courses
PRODUCT_ID: 15,9887
Cluster 6
Count: 15059
Percent: 13,69%
CATEGORY: Drinks
PRODUCT_ID: 2,0080
Cluster 7
Count: 29829
Percent: 27,13%
CATEGORY: Salads
PRODUCT_ID: 6,4930
Cluster 5
Count: 10099
Percent: 9,18%
CATEGORY: Soups
PRODUCT_ID: 10,5053
Cluster 4
Count: 54981
Percent: 50%
CATEGORY: Main courses
PRODUCT_ID: 16,9959
Cluster 11
Count: 9976
Percent: 9,07%
CATEGORY: Salads
PRODUCT_ID: 8,4902
Cluster 10
Count: 19853
Percent: 18,05%
CATEGORY: Salads
PRODUCT_ID: 5,4894
Cluster 9
Count: 9972
Percent: 9,07%
CATEGORY: Soups
PRODUCT_ID: 12,5051
Cluster 8
Count: 45009
Percent: 40,93%
CATEGORY: Main courses
PRODUCT_ID: 17,9909
Cluster 13
Count: 9876
Percent: 8,98%
CATEGORY: Salads
PRODUCT_ID: 6,4883
Cluster 12
Count: 9977
Percent: 9,07%
PRODUCT_ID: 4,5008
CATEGORY: Salads
Cluster 15
Count: 10022
Percent: 9,11%
PRODUCT_ID: 17,4934
CATEGORY: Main courses
Cluster 14
Count: 34987
Percent: 31,82%
CATEGORY: Main courses
PRODUCT_ID: 18,1334
Cluster 16
Count: 10133
Percent: 9,21%
PRODUCT_ID: 15,5005
CATEGORY: Main courses
Cluster 17
Count: 24854
Percent: 22,6%
CATEGORY: Main courses
PRODUCT_ID: 19,2068
Cluster 19
Count: 9964
Percent: 9,06%
PRODUCT_ID: 21,5115
CATEGORY: Main courses
Cluster 18
Count: 14890
Percent: 13,54%
CATEGORY: Main courses
PRODUCT_ID: 17,6645


Cluster 1
Count: 109968
Percent: 100%
CATEGORY: Main courses
RATING: GOOD
PRODUCT_ID: 11,4985
ID: 144 996,5507
Cluster 2
Count: 45009
Percent: 40,93%
CATEGORY: Main courses
RATING: GOOD
ID: 145 114,6053
PRODUCT_ID: 17,9909
Cluster 3
Count: 64959
Percent: 59,07%
CATEGORY: Salads
RATING: BAD
ID: 144 914,7527
PRODUCT_ID: 7,0000
Cluster 4
Count: 18837
Percent: 17,13%
CATEGORY: Main courses
RATING: GOOD
PRODUCT_ID: 17,9887
ID: 113 253,1724
Cluster 5
Count: 26172
Percent: 23,8%
CATEGORY: Main courses
RATING: BAD
ID: 168 046,5100
PRODUCT_ID: 17,9925
Cluster 7
Count: 53189
Percent: 48,37%
CATEGORY: Salads
RATING: BAD
ID: 139 894,0349
PRODUCT_ID: 7,9950
Cluster 6
Count: 11770
Percent: 10,7%
CATEGORY: Drinks
RATING: GOOD
ID: 167 603,5343
PRODUCT_ID: 2,5037
Cluster 17
Count: 13173
Percent: 11,98%
RATING: BAD
ID: 167 791,8319
PRODUCT_ID: 17,9869
CATEGORY: Main courses
Cluster 16
Count: 12999
Percent: 11,82%
RATING: GOOD
ID: 168 304,5972
PRODUCT_ID: 17,9982
CATEGORY: Main courses
Cluster 9
Count: 16130
Percent: 14,67%
CATEGORY: Salads
RATING: BAD
PRODUCT_ID: 9,0918
ID: 180 253,4911
Cluster 8
Count: 37059
Percent: 33,7%
CATEGORY: Salads
RATING: BAD
ID: 122 327,5051
PRODUCT_ID: 7,5176
Cluster 19
Count: 5801
Percent: 5,28%
CATEGORY: Drinks
ID: 167 780,3846
PRODUCT_ID: 2,4959
RATING: BAD
Cluster 18
Count: 5969
Percent: 5,43%
CATEGORY: Drinks
ID: 167 431,6616
PRODUCT_ID: 2,5111
RATING: GOOD
Cluster 11
Count: 8270
Percent: 7,52%
CATEGORY: Drinks
RATING: BAD
ID: 112 549,4631
PRODUCT_ID: 2,5024
Cluster 10
Count: 28789
Percent: 26,18%
CATEGORY: Salads
RATING: BAD
ID: 125 136,3699
PRODUCT_ID: 8,9583
Cluster 12
Count: 12576
Percent: 11,44%
CATEGORY: Soups
RATING: BAD
ID: 124 483,7688
PRODUCT_ID: 11,5272
Cluster 13
Count: 16213
Percent: 14,74%
CATEGORY: Salads
RATING: GOOD
ID: 125 642,5755
PRODUCT_ID: 6,9656
Cluster 15
Count: 8146
Percent: 7,41%
RATING: GOOD
ID: 126 223,0563
PRODUCT_ID: 6,9487
CATEGORY: Salads
Cluster 14
Count: 8067
Percent: 7,34%
RATING: BAD
ID: 125 056,4099
PRODUCT_ID: 6,9828
CATEGORY: Salads



Dzięki uzyskanym wynikom udało nam się zidentyfikować jakie grupy produktów powinny być rozwijane w naszych restauracjach. Pokazało jakie produkty nie cieszą się dobrą opinią i powinny zostać ulepszone, aby podnieść jakość dań w naszej restauracji.

7. Wnioski

Projekt pozwolił zbudować hurtowanie danych oraz zapoznać się z pracą z nimi. Mogliśmy poznać jak optymalnie stworzyć hurtownię, aby gromadzone dane były przydatne dla biznesu.

Stworzone zapytania pozwoliły nam się zaznajomić z zaawansowanymi konceptami jakie możemy spotkać w bazach danych Oracle SQL. Pokazuje nam to jakie możliwości raportowania możemy osiągnąć samym kodem bez zewnętrznych aplikacji. Ze względu na losowość danych nie niosą one zbyt wiele informacji.

Poznany moduł data miner pozwolił nam spojrzeć na to jak dane są przetwarzane oraz jakie zależności można wyłapać. Dzięki temu wiemy, jak tworzyć takie modele oraz używać ich w praktyce.