

## > IMPORTS


↳ 2 cells hidden

### ✓ Exploration de la base de données

### ✓ Observation et mise en place

```
#Lire le fichier excel  
df = pd.read_excel("all_data.xlsx", engine="openpyxl")
```

```
# Filtrer les lignes où Chip_type == "ISAC_V1"  
ALEX_df = df[df["Chip_Type"] == "ALEX"]  
ALEX_df.head()
```



	Unnamed: 0	Chip_Code	Chip_Type	Chip_Image_Name	Age	Gender	Blood_Mor
1379	PMP0237	02AGT832	ALEX	NaN	28.0	F	
1380	PMP0238	02AGT834	ALEX	NaN	20.0	M	
1381	PMP0239	02AGT835	ALEX	NaN	22.0	F	
1382	PMP0240	02AGT486	ALEX	NaN	10.0	M	
1383	PMP0241	02AGT488	ALEX	NaN	2.0	M	

5 rows × 386 columns

```
ALEX_df.describe()
```



	Age	Blood_Month_sample	French_Residence_Department	French_Re
<b>count</b>	1139.000000	1075.000000	1139.000000	1139.0
<b>mean</b>	26.849868	6.299535	447.201054	17.2
<b>std</b>	19.376333	3.097057	472.271145	33.1
<b>min</b>	0.000000	1.000000	1.000000	1.0
<b>25%</b>	11.000000	4.000000	38.000000	1.0
<b>50%</b>	22.000000	7.000000	69.000000	2.0
<b>75%</b>	40.000000	8.000000	999.000000	9.0
<b>max</b>	85.000000	12.000000	999.000000	99.0

8 rows × 366 columns

Comme on peut le voir, il y a beaucoup de valeurs manquantes. C'est pourquoi nous supprimons les colonnes contenant des allergènes sans valeur.

```
ALEX= ALEX_df.dropna(axis=1)
ALEX.describe()
```



	Age	French_Residence_Department	French_Region	Rural_or_urban
<b>count</b>	1139.000000	1139.000000	1139.000000	1139.
<b>mean</b>	26.849868	447.201054	17.287094	7.
<b>std</b>	19.376333	472.271145	33.100942	3.
<b>min</b>	0.000000	1.000000	1.000000	0.
<b>25%</b>	11.000000	38.000000	1.000000	9.
<b>50%</b>	22.000000	69.000000	2.000000	9.
<b>75%</b>	40.000000	999.000000	9.000000	9.
<b>max</b>	85.000000	999.000000	99.000000	9.

8 rows × 347 columns

## ✓ Ingenierie des données et reformulation

```
from tqdm import tqdm
import pandas as pd

def expand_custom_one_hot(df, value_lists: dict):
    df = df.copy()
    for col in tqdm(value_lists.keys()):
        valid_values = value_lists[col]

        for val in valid_values:
            df.loc[:, f"{col}_{val}"] = 0

        split_values = df[col].astype(str).str.split(r"[.]", expand=True)

        # remplir les colonnes
        for idx, row in split_values.iterrows():
            for val in row:
                if val and val.strip().isdigit():
                    val_int = int(val.strip())
                    if val_int in valid_values:
                        df.loc[idx, f"{col}_{val_int}"] = 1

    return df
```

```

value_lists = {
    'Treatment_of_rhinitis': [0, 1, 2, 3, 4, 9],
    'Treatment_of_athsma': [0, 1, 2, 3, 4, 5, 6, 7, 9, 10, 11],
    'General_cofactors': [0,1,2,3,4,5,6,7,8,9,10,11,12],
    'Age_of_onsets': [0,1,2,3,4,5,6,9],
    'Treatment_of_atopic_dematitis': [0,1,2,3,4,5,6,9],
    'ARIA_(rhinitis)': [0,1,2,3,4,5,9],
    'GINA_(asthma)': [0,1,2,3,4,5,9]
}
ALEX= expand_custom_one_hot(ALEX, value_lists)
ALEX = ALEX.drop('Type_of_Respiratory_Allergy', axis=1)
ALEX = ALEX.drop('Type_of_Food_Allergy', axis=1)
ALEX = ALEX.drop('Type_of_Venom_Allergy', axis=1)
ALEX = ALEX.drop(columns=[
    'Treatment_of_rhinitis',
    'Treatment_of_athsma',
    'General_cofactors',
    'Age_of_onsets',
    'Treatment_of_atopic_dematitis',
    'ARIA_(rhinitis)',
    'GINA_(asthma)',
    'Food_List',
    'Oral_food_challenge',
    'Symptoms_per_food'
])

```

 [Show hidden output](#)

```
ALEX.to_excel('ALEX.xlsx', index=False)
```

Ne pas executer cette commande ( les models SVM, et Logistic regression ne fonctionne pas sur des NAN)

```

"""
V1[[
    'Rural_or_urban_area', 'Skin_Symptoms', 'Conjunctivitis', 'Oral_Syndrom',
    'Cardiovascular_symptoms', 'Respiratory_symptoms', 'Gastrointestinal_symptc
    'History_of_food_anaphylaxis', 'First_degree_family_history_of_atopy',
    'History_of_atopic_dematitis', 'History_of_mast_cell_disorders',
    'History_of_hymenoptera_venom_anaphylaxis', 'Mammalian_meat', 'Food_polyall
]] = V1[[
    'Rural_or_urban_area', 'Skin_Symptoms', 'Conjunctivitis', 'Oral_Syndrom',
    'Cardiovascular_symptoms', 'Respiratory_symptoms', 'Gastrointestinal_symptc
    'History_of_food_anaphylaxis', 'First_degree_family_history_of_atopy',
    'History_of_atopic_dematitis', 'History_of_mast_cell_disorders',
    'History_of_hymenoptera_venom_anaphylaxis', 'Mammalian_meat', 'Food_polyall
]].replace({9: np.nan})

```

```

"""

import pandas as pd

# categories des allergenes
import pandas as pd

# Dictionnaire des catégories d'allergènes
allergenes = {
    "Arbres et Pollens": [
        "Aln_g_1", "Aln_g_4", "Bet_v_1", "Bet_v_2", "Bet_v_6", "Cor_a_1.0401",
        "Cor_a_1.0103", "Cor_a_11", "Cor_a_12_RU0", "Cor_a_pollen", "Cor_a_14",
        "Ole_e_1", "Ole_e_7_RU0", "Ole_e_9", "Pla_a_1", "Pla_a_2", "Pla_a_3",
        "Cup_a_1", "Cup_s", "Cyp_c_1", "Jun_a", "Act_d_1", "Art_v_1", "Art_v_3",
        "Fag_e_2", "Jug_r_1", "Jug_r_2", "Jug_r_3", "Jug_r_4", "Jug_r_6",
        "Jug_r_pollen", "Sal_k_1", "Thu_a_1"
    ],
    "Graminées et Herbacées": [
        "Cyn_d_1", "Lol_p_1", "Phl_p_1", "Phl_p_12", "Phl_p_2", "Phl_p_5.0101",
        "Phl_p_6", "Phl_p_7", "Amb_a_1", "Amb_a_4", "Che_a_1", "Par_j_2"
    ],
    "Acariens": [
        "Der_f_1", "Der_f_2", "Der_p_1", "Der_p_10", "Der_p_11", "Der_p_2",
        "Der_p_20", "Der_p_21", "Der_p_23", "Der_p_5", "Der_p_7", "Blo_t_5",
        "Blo_t_10", "Blo_t_21", "Lep_d_2"
    ],
    "Animaux": [
        "Fel_d_1", "Fel_d_2", "Fel_d_4", "Fel_d_7", "Can_f_1", "Can_f_2",
        "Can_f_3", "Can_f_4", "Can_f_6", "Can_f_Fd1", "Can_f_male_urine",

```

```

    "Equ_c_1", "Equ_c_3", "Equ_c_4", "Equ_c_meat", "Equ_c_milk", "Bla_g_1",
    "Bla_g_2", "Bla_g_4", "Bla_g_5", "Bla_g_9", "Bos_d_2", "Bos_d_4",
    "Bos_d_5", "Bos_d_6", "Bos_d_8", "Bos_d_meat", "Bos_d_milk", "Mus_m_1",
    "Rat_n"
],

"Aliments Végétaux": [
    "Act_d_1", "Ana_o_2", "Ana_o_3", "Mal_d_1", "Mal_d_2", "Mal_d_3",
    "Pru_p_3", "Pru_p_7_RU0", "Ara_h_1", "Ara_h_2", "Ara_h_3", "Ara_h_6",
    "Ara_h_8", "Ara_h_9", "Ara_h_15", "Gly_m_4", "Gly_m_5", "Gly_m_6",
    "Gly_m_8", "Pis_v_1", "Pis_v_2", "Pis_v_3", "Ber_e_1", "Cor_a_8",
    "Cor_a_9", "Ses_i_1", "Tri_a_14", "Tri_a_aA_TI", "Tri_a_19", "Hor_v",
    "Sec_c_flour"
],

"Aliments Animaux": [
    "Bos_d_milk", "Gal_d_1", "Gal_d_2", "Gal_d_3", "Gal_d_4", "Gal_d_5",
    "Gal_d_meat", "Gal_d_white", "Gal_d_yolk", "Gad_m_1", "Gad_m_2+3",
    "Sal_s_1", "Thu_a_1", "Myt_e"
],

"Moisissures": [
    "Asp_f_1", "Asp_f_3", "Asp_f_4", "Asp_f_6", "Alt_a_1", "Alt_a_6",
    "Cla_h_8", "Cla_h", "Pen_m_1", "Pen_m_2", "Pen_m_3", "Pen_m_4", "Pen_ct
],

"Divers": [
    "Hev_b_1", "Hev_b_3", "Hev_b_5", "Hev_b_6.02", "Hev_b_8", "Hev_b_11",
    "Ves_v_5", "Ves_v_1", "Pol_d_5", "Ani_s_1", "Ani_s_3", "Xip_g_1"
],

"Autres": [
    "Ail_a", "Fra_a_1+3", "Zea_m", "Ory_c_1", "Sus_d_1"
]
}

```

#mettre les colonnes min et max et moyennes

```

for type_allergene, colonnes in allergenes.items():
    colonnes_presentes = [col for col in colonnes if col in ALEX.columns]
    ALEX[f"Moyenne_{type_allergene}"] = ALEX[colonnes_presentes].mean(axis=1)
    ALEX[f"Max_{type_allergene}"] = ALEX[colonnes_presentes].max(axis=1)
    ALEX[f"Min_{type_allergene}"] = ALEX[colonnes_presentes].min(axis=1)

```

ALEX.describe()



	Age	French_Residence_Department	French_Region	Rural_or_urban
count	1139.000000	1139.000000	1139.000000	1139.
mean	26.849868	447.201054	17.287094	7.
std	19.376333	472.271145	33.100942	3.
min	0.000000	1.000000	1.000000	0.
25%	11.000000	38.000000	1.000000	9.
50%	22.000000	69.000000	2.000000	9.
75%	40.000000	999.000000	9.000000	9.
max	85.000000	999.000000	99.000000	9.

8 rows x 431 columns

```

col_allergenes = [
    "Act_d_1", "Act_d_2", "Act_d_5", "Aln_g_1", "Alt_a_1", "Alt_a_6", "Amb_a_1"
    "Ana_o_2", "Ani_s_1", "Ani_s_3", "Api_g_1", "Api_m_1", "Ara_h_1", "Ara_h_2"
    "Ara_h_3", "Ara_h_6", "Ara_h_8", "Ara_h_9", "Art_v_1", "Art_v_3", "Asp_f_1"
    "Asp_f_3", "Asp_f_6", "Ber_e_1", "Bet_v_1", "Bet_v_2", "Bla_g_1", "Bla_g_2"
    "Bla_g_5", "Blo_t_5", "Bos_d_4", "Bos_d_5", "Bos_d_6", "Bos_d_8", "Can_f_1"
    "Can_f_2", "Can_f_3", "Che_a_1", "Cla_h_8", "Cor_a_1.0401", "Cor_a_8", "Cor
    "Cry_j_1", "Cup_a_1", "Cyn_d_1", "Der_f_1", "Der_f_2", "Der_p_1", "Der_p_10"
    "Der_p_2", "Equ_c_1", "Equ_c_3", "Fag_e_2", "Fel_d_1", "Fel_d_2", "Fel_d_4"
    "Gal_d_1", "Gal_d_2", "Gal_d_3", "Gal_d_5", "Gly_m_4", "Gly_m_5", "Gly_m_6"
    "Hev_b_1", "Hev_b_3", "Hev_b_5", "Hev_b_8", "Jug_r_1", "Jug_r_2", "Jug_r_3"
    "Lep_d_2", "Mal_d_1", "Mer_a_1", "Mus_m_1", "Ole_e_1", "Ole_e_9", "Par_j_2"
    "Pen_m_1", "Pen_m_2", "Pen_m_4", "Phl_p_1", "Phl_p_12", "Phl_p_2", "Phl_p_6"
    "Phl_p_7", "Pla_a_1", "Pla_a_2", "Pla_a_3", "Pla_l_1", "Pol_d_5", "Pru_p_3"
    "Sal_k_1", "Ses_i_1", "Tri_a_14", "Tri_a_aA_TI", "Ves_v_5", "Cor_a_14", "Ca
    "Der_p_23", "Ana_o_3", "Can_f_6", "Aca_m", "Aca_s", "Ach_d", "Act_d_10", "A
    "All_c", "All_s", "Aln_g_4", "Ama_r", "Amb_a", "Amb_a_4", "Ana_o", "Api_g_2"
    "Api_g_6", "Api_m", "Api_m_10", "Ara_h_15", "Arg_r_1", "Art_v", "Asp_f_4",
    "Ave_s", "Ber_e", "Bet_v_6", "Bla_g_4", "Bla_g_9", "Blo_t_10", "Blo_t_21",
    "Bos_d_2", "Bos_d_meat", "Bos_d_milk", "Bro_pa", "Cam_d", "Can_f_Fd1",
    "Can_f_male_urine", "Can_s", "Can_s_3", "Cap_a", "Cap_h_epithelia", "Cap_h
    "Car_c", "Car_i", "Car_p", "Cav_p_1", "Che_a", "Che_q", "Chi_spp.", "Cic_a"
    "Cit_s", "Cla_h", "Clu_h", "Clu_h_1", "Cor_a_1.0103", "Cor_a_11", "Cor_a_12
    "Cor_a_pollen", "Cra_c_6", "Cuc_m_2", "Cuc_p", "Cup_s", "Cyn_d", "Cyp_c_1",
    "Dau_c", "Dau_c_1", "Der_p_11", "Der_p_20", "Der_p_21", "Der_p_5", "Der_p_7
    "Dol_spp", "Equ_c_4", "Equ_c_meat", "Equ_c_milk", "Fag_e", "Fag_s_1", "Fel
    "Fic_b", "Fic_c", "Fra_a_1+3", "Fra_e", "Fra_e_1", "Gad_m", "Gad_m_1", "Gac
    "Gal_d_4", "Gal_d_meat", "Gal_d_white", "Gal_d_yolk", "Gly_d_2", "Gly_m_8",
    "Hel_a", "Hev_b_11", "Hev_b_6.02", "Hom_g", "Hom_s_LF", "Hor_v", "Jug_r_4",
    "Jug_r_6", "Jug_r_pollen", "Jun_a", "Len_c", "Lit_s", "Loc_m", "Lol_p_1",
    "Lol_spp.", "Lup_a", "Mac_i_2S_Albumin", "Mac_inte", "Mal_d_2", "Mal_d_3",
    "Mala_s_11", "Mala_s_5", "Mala_s_6", "Man_i", "Mel_g", "Mes_a_1_RU0", "Mor
    "Mus_a", "Myt_e", "Ole_e_7_RU0", "Ori_v", "Ory_c_1", "Ory_c_2", "Ory_c_3",
    "Ory_s", "Ory_meat", "Ost_e", "Ovi_a_epithelia", "Ovi_a_meat", "Ovi_a_milk"
    "Pan_b", "Pan_m", "Pap_s", "Pap_s_2S_Albumin", "Par_j", "Pas_n", "Pec_spp."
    "Pen_ch", "Pen_m_3", "Per_a", "Per_a_7", "Pers_a", "Pet_c", "Pha_v", "Phl_p
    "Pho_d_2", "Phod_s_1", "Phr_c", "Pim_a", "Pis_s", "Pis_v_1", "Pis_v_2", "Pi
    "Pis_v_4_RU0", "Pla_l", "Pol_d", "Pop_n", "Pru_av", "Pru_du", "Pru_p_7_RU0"
    "Pyr_c", "Raj_c", "Raj_c_Parvalbumin", "Rat_n", "Rud_spp.", "Sac_c", "Sal_k
    "Sal_s", "Sal_s_1", "Sco_s", "Sco_s_1", "Sec_c_flour", "Sec_c_pollen", "Ses
    "Sin", "Sin_a_1", "Sol_spp.", "Sol_t", "Sola_l", "Sola_l_6", "Sus_d_1",
    "Sus_d_epithelia", "Sus_d_meat", "Ten_m", "Thu_a", "Thu_a_1", "Tri_a_19",
    "Tri_fo", "Tri_s", "Tyr_p", "Tyr_p_2", "Ulm_c", "Urt_d", "Vac_m", "Ves_v",
    "Ves_v_1", "Vit_v_1", "Xip_g_1", "Zea_m", "Zea_m_14"
]
ALEX.drop(col_allergenes, axis=1, inplace=True)

```




Il y a des valeurs ou Allergy\_Present = 0 et d'autre Type d'allergy sont presente c pour cela j'ai nettoyer en mettant tous les autres types d'allergie egale à 0 si Allergy\_Present = 0

```
ALEX.loc[ALEX["Allergy_Present"] == 0, ["Respiratory_Allergy", "Food_Allergy",
ALEX.loc[ALEX["Respiratory_Allergy"] == 0, [
    "Type_of_Respiratory_Allergy_IGE_Pollen_Herb",
    "Type_of_Respiratory_Allergy_IGE_Pollen_Tree",
    "Type_of_Respiratory_Allergy_IGE_Dander_Animals",
    "Type_of_Respiratory_Allergy_IGE_Mite_Cockroach",
    "Type_of_Respiratory_Allergy_IGE_Molds_Yeast",
    "Type_of_Respiratory_Allergy_ARIA",
    "Type_of_Respiratory_Allergy_CONJ",
    "Type_of_Respiratory_Allergy_IGE_Pollen_Gram",
    "Type_of_Respiratory_Allergy_GINA"
]] = 0
```

```
ALEX.loc[ALEX["Food_Allergy"] == 0, [
    "Type_of_Food_Allergy_Aromatics",
    "Type_of_Food_Allergy_Cereals_&_Seeds",
    "Type_of_Food_Allergy_Egg",
    "Type_of_Food_Allergy_Fish",
    "Type_of_Food_Allergy_Fruits_and_Vegetables",
    "Type_of_Food_Allergy_Mammalian_Milk",
    "Type_of_Food_Allergy_Oral_Syndrom",
    "Type_of_Food_Allergy_Other_Legumes",
    "Type_of_Food_Allergy_Peanut",
    "Type_of_Food_Allergy_Shellfish",
    "Type_of_Food_Allergy_TP0",
    "Type_of_Food_Allergy_Tree_Nuts"
]] = 0
```

```
ALEX.loc[ALEX["Venom_Allergy"] == 0, ["Type_of_Venom_Allergy_ATCD_Venom",
    "Type_of_Venom_Allergy_IGE_Venom"]] = 0
```

```
ALEX.describe()
```



	Age	French_Residence_Department	French_Region	Rural_or_urban
count	1139.000000	1139.000000	1139.000000	1139.
mean	26.849868	447.201054	17.287094	7.
std	19.376333	472.271145	33.100942	3.
min	0.000000	1.000000	1.000000	0.
25%	11.000000	38.000000	1.000000	9.
50%	22.000000	69.000000	2.000000	9.
75%	40.000000	999.000000	9.000000	9.
max	85.000000	999.000000	99.000000	9.

8 rows x 131 columns

```
ALEX.to_excel('ALEX.xlsx', index=False)
```

## ✓ Repartition suivant chaque Target

```
import pandas as pd
import plotly.express as px

value_counts = ALEX["Allergy_Present"].value_counts(normalize=True) * 100
df_plot = value_counts.reset_index()
df_plot.columns = ["Classe", "Pourcentage"]
df_plot["Classe"] = df_plot["Classe"].astype(str)

# Remplacer les valeurs 0/1 par des libellés explicites
classe_labels = {"0": "Sans allergie", "1": "Avec allergie"}
df_plot["Classe"] = df_plot["Classe"].map(classe_labels)

fig = px.pie(
    df_plot,
    names="Classe",
    values="Pourcentage",
    title="Répartition des classes dans 'Allergy_Present'",
    color="Classe",
    color_discrete_map={
```

```

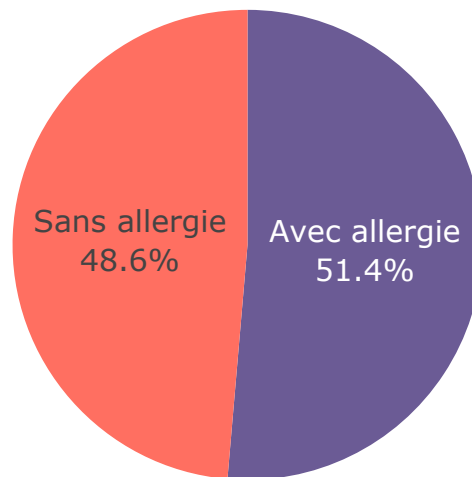
        "Sans allergie": "#FF6F61",
        "Avec allergie": "#6B5B95"
    }
)

fig.update_traces(textinfo='percent+label', textfont_size=14)
fig.update_layout(
    title_font_size=15,
    width=800,
    height=400
)
fig.show()

```



Répartition des classes dans 'Allergy\_Present'



```

import pandas as pd
import plotly.graph_objects as go
from plotly.subplots import make_subplots

allergy_positive = ALEX[ALEX["Allergy_Present"] == 1]

targets = ["Respiratory_Allergy", "Food_Allergy", "Venom_Allergy"]
labels_map = {"0": "Sans allergie", "1": "Avec allergie"}
colors_map = {
    "Sans allergie": "#FF6F61",
    "Avec allergie": "#6B5B95"
}

```

```
fig = make_subplots(rows=1, cols=3, specs=[[{'type':'domain'}]*3],
                    subplot_titles=targets)

for i, col in enumerate(targets):
    counts = ALEX[col].value_counts(normalize=True) * 100
    df_plot = counts.reset_index()
    df_plot.columns = ["Classe", "Pourcentage"]
    df_plot["Classe"] = df_plot["Classe"].astype(str).map(labels_map)

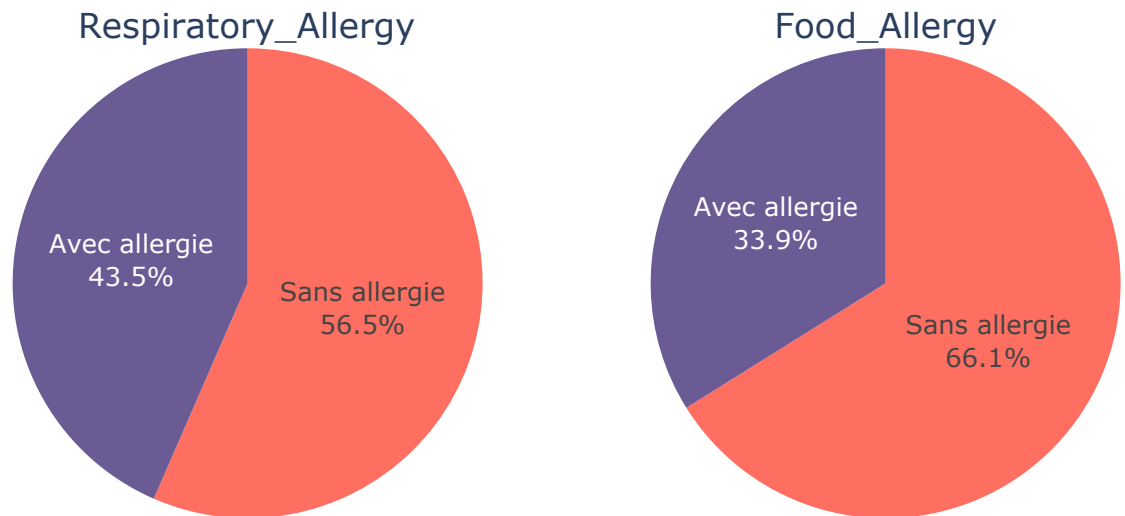
    fig.add_trace(
        go.Pie(
            labels=df_plot["Classe"],
            values=df_plot["Pourcentage"],
            name=col,
            marker=dict(colors=[colors_map[label] for label in df_plot["Classe"]
                                textinfo="percent+label"
            ),
            row=1, col=i+1
        )

fig.update_layout(
    title_text="Répartition des allergies par type",
    title_font_size=18,
    height=400,
    width=1000,
    showlegend=False
)

fig.show()
```



## Répartition des allergies par type



```
import pandas as pd
import plotly.express as px

res_pos = ALEX[ALEX["Respiratory_Allergy"] == 1]
food_pos = ALEX[ALEX["Food_Allergy"] == 1]
venom_pos = ALEX[ALEX["Venom_Allergy"] == 1]

columns = [
    "Type_of_Respiratory_Allergy_IGE_Pollen_Herb",
    "Type_of_Respiratory_Allergy_IGE_Pollen_Tree",
    "Type_of_Respiratory_Allergy_IGE_Dander_Animals",
    "Type_of_Respiratory_Allergy_IGE_Mite_Cockroach",
    "Type_of_Respiratory_Allergy_IGE_Molds_Yeast",
    "Type_of_Respiratory_Allergy_ARIA",
    "Type_of_Respiratory_Allergy_CONJ",
    "Type_of_Respiratory_Allergy_IGE_Pollen_Gram",
    "Type_of_Respiratory_Allergy_GINA"
]

label_map = {"0": "Non allergique", "1": "Allergique"}
color_map = {"Non allergique": "#FF6F61", "Allergique": "#6B5B95"}

for col in columns:
    counts = res_pos[col].value_counts(normalize=True) * 100
    df_plot = counts.reset_index()
```

```

df_plot.columns = ["Classe", "Pourcentage"]
df_plot["Classe"] = df_plot["Classe"].astype(str).map(label_map)

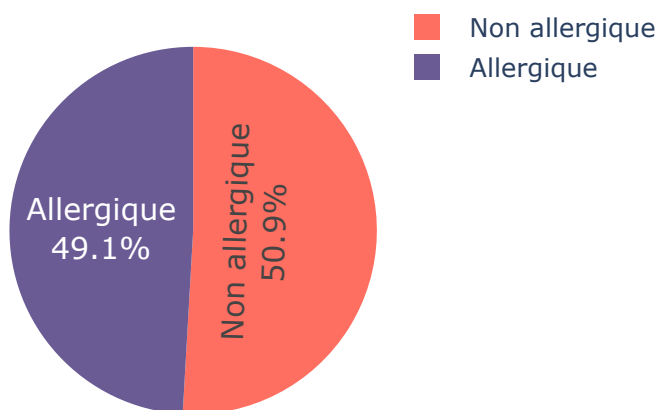
fig = px.pie(
    df_plot,
    names="Classe",
    values="Pourcentage",
    title=col,
    color="Classe",
    color_discrete_map=color_map
)

fig.update_traces(textinfo='percent+label', textfont_size=14)
fig.update_layout(
    title_font_size=16,
    width=400,
    height=400,
    showlegend=True
)
fig.show()

```

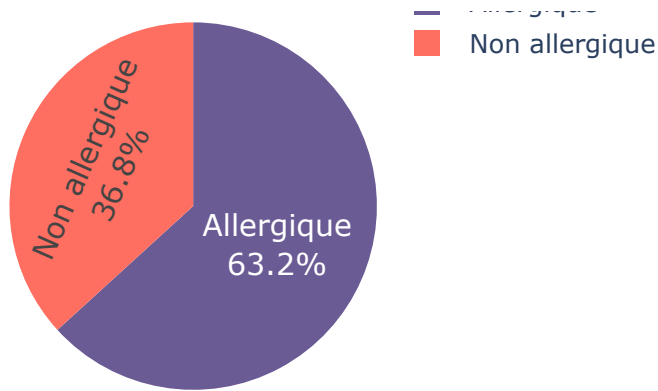


### Type\_of\_Respiratory\_Allergy\_IGE\_Pollen\_Herb

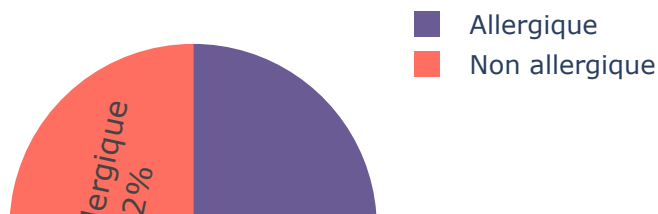


### Type\_of\_Respiratory\_Allergy\_IGE\_Pollen\_Tree

■ Alleraiaue



### Type\_of\_Respiratory\_Allergy\_IGE\_Dander\_Anir



```

food_subtypes = [
    "Type_of_Food_Allergy_Aromatics",
    "Type_of_Food_Allergy_Cereals_&_Seeds",
    "Type_of_Food_Allergy_Egg",
    "Type_of_Food_Allergy_Fish",
    "Type_of_Food_Allergy_Fruits_and_Vegetables",
    "Type_of_Food_Allergy_Mammalian_Milk",
    "Type_of_Food_Allergy_Oral_Syndrom",
    "Type_of_Food_Allergy_Other_Legumes",
    "Type_of_Food_Allergy_Peanut",
    "Type_of_Food_Allergy_Shellfish",
    "Type_of_Food_Allergy_TPO",
    "Type_of_Food_Allergy_Tree_Nuts"
]

label_map = {"0": "Non allergique", "1": "Allergique"}
color_map = {"Non allergique": "#FF6F61", "Allergique": "#6B5B95"}

for col in food_subtypes:
    counts = food_pos[col].value_counts(normalize=True) * 100
    df_plot = counts.reset_index()
  
```

```

df_plot.columns = ["Classe", "Pourcentage"]
df_plot["Classe"] = df_plot["Classe"].astype(str).map(label_map)

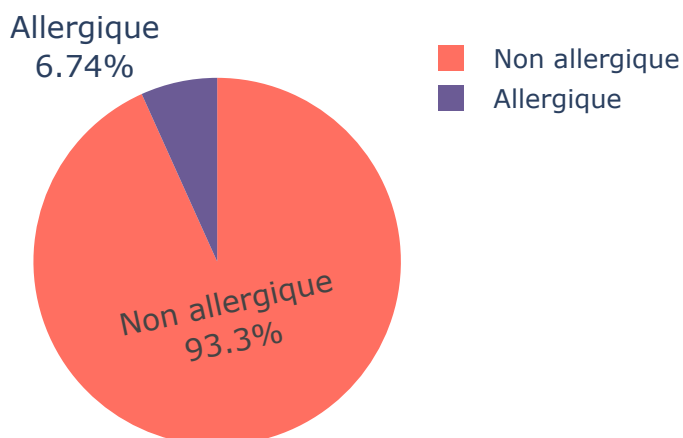
fig = px.pie(
    df_plot,
    names="Classe",
    values="Pourcentage",
    title=col,
    color="Classe",
    color_discrete_map=color_map
)

fig.update_traces(textinfo='percent+label', textfont_size=14)
fig.update_layout(
    title_font_size=16,
    width=400,
    height=400,
    showlegend=True
)
fig.show()

```



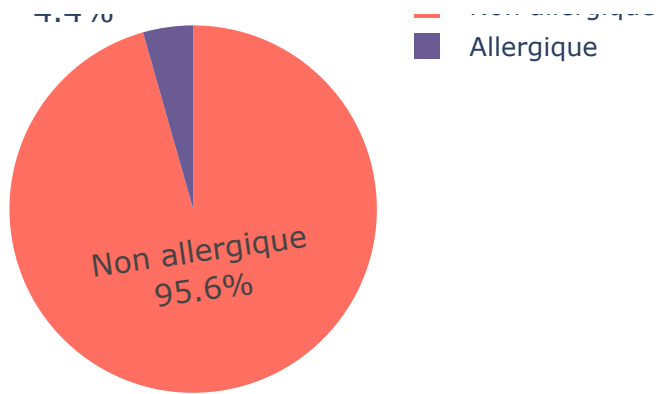
### Type\_of\_Food\_Allergy\_Aromatics



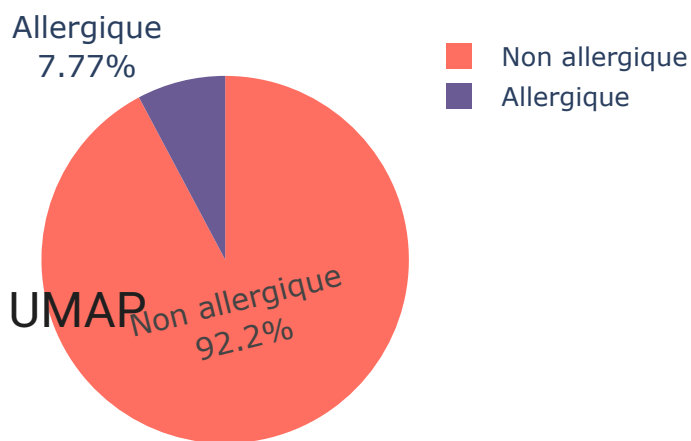
### Type\_of\_Food\_Allergy\_Cereals\_&\_Seeds





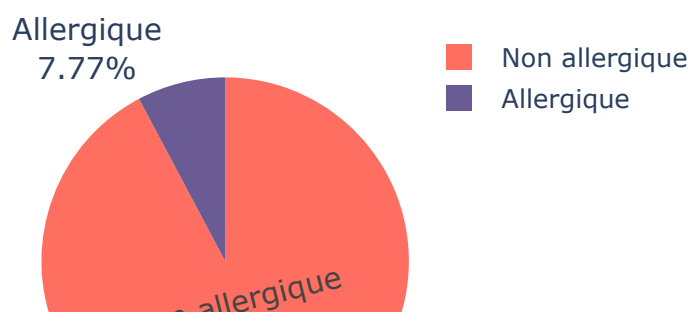


### Type\_of\_Food\_Allergy\_Egg



✓ PCA et UMAP

### Type\_of\_Food\_Allergy\_Fish



```


targets = [
    "Allergy_Present",
    "Respiratory_Allergy",
    "Food_Allergy",
    "Venom_Allergy",
    "Severe_Allergy",
    "Type_of_Respiratory_Allergy_IGE_Pollen_Herb",
    "Type_of_Respiratory_Allergy_IGE_Pollen_Tree",
    "Type_of_Respiratory_Allergy_IGE_Dander_Animals",
    "Type_of_Respiratory_Allergy_IGE_Mite_Cockroach",
    "Type_of_Respiratory_Allergy_IGE_Molds_Yeast",
    "Type_of_Food_Allergy_Aromatics",
    "Type_of_Food_Allergy_Other",
    "Type_of_Food_Allergy_Cereals_&_Seeds",
    "Type_of_Food_Allergy_Egg",
    "Type_of_Food_Allergy_Fish",
    "Type_of_Food_Allergy_Fruits_and_Vegetables",
    "Type_of_Food_Allergy_Mammalian_Milk",
    "Type_of_Food_Allergy_Oral_Syndrom",
    "Type_of_Food_Allergy_Other_Legumes",
    "Type_of_Food_Allergy_Peanut",
    "Type_of_Food_Allergy_Shellfish",
    "Type_of_Food_Allergy_TPO",
    "Type_of_Food_Allergy_Tree_Nuts",
    "Type_of_Venom_Allergy_ATCD_Venom",
    "Type_of_Venom_Allergy_IGE_Venom",

]

extra_columns = [
    "Chip_Type",
    "Chip_Code",
    "Gender"
]

extra = ['History_of_food_anaphylaxis', 'First_degree_family_history_of_atopy',
        'History_of_hymenoptera_venom_anaphylaxis', 'Mammalian_meat']

```



```

from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import pandas as pd

X= ALEX.copy()
X.drop(targets, axis=1, inplace=True)
X.drop(extra_columns, axis=1, inplace=True)
X.drop(extra, axis=1, inplace=True)

```

```

X = X.iloc[:, 1:]

y = ALEX["Allergy_Present"]

# 3. Standardisation
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 4. PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Reconstruction d'un DataFrame avec les résultats PCA
df_pca = pd.DataFrame(X_pca, columns=["PC1", "PC2"])
df_pca["Allergy_Present"] = y.values # Ajouter la cible

# Tracé du graphique
fig = px.scatter(
    df_pca,
    x="PC1",
    y="PC2",
    color=df_pca["Allergy_Present"].astype(str), # couleur par classe 0 ou 1
    title="Projection PCA des patients",
    labels={
        "PC1": "Première composante principale",
        "PC2": "Deuxième composante principale",
        "color": "Présence d'allergie"
    },
    color_discrete_map={"0": "blue", "1": "red"},
    width=900,
    height=650
)

fig.update_layout(
    legend_title="Présence d'allergie"
)

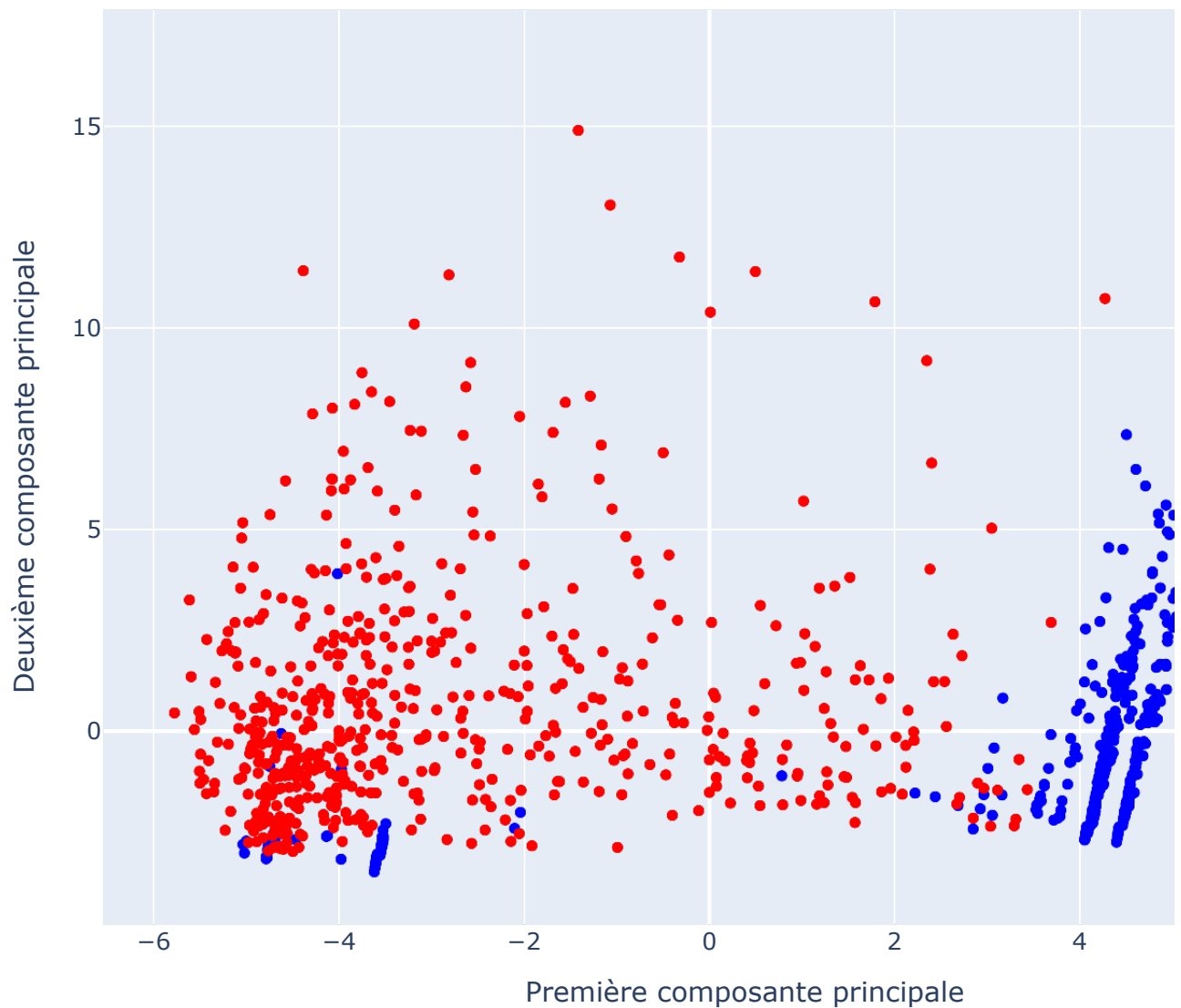
fig.show()

```

Type\_of\_Food\_Allergy\_Peanut



## Projection PCA des patients



```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import pandas as pd
import plotly.express as px
```

# 1. Préparation des données

```
X = ALEX.copy()
X.drop(targets, axis=1, inplace=True)
X.drop(extra_columns, axis=1, inplace=True)
X.drop(extra, axis=1, inplace=True)
X = X.iloc[:, 1:]
```

```
y = ALEX["Allergy_Present"]

# 2. Standardisation
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 3. PCA en 3D
pca = PCA(n_components=3)
X_pca = pca.fit_transform(X_scaled)

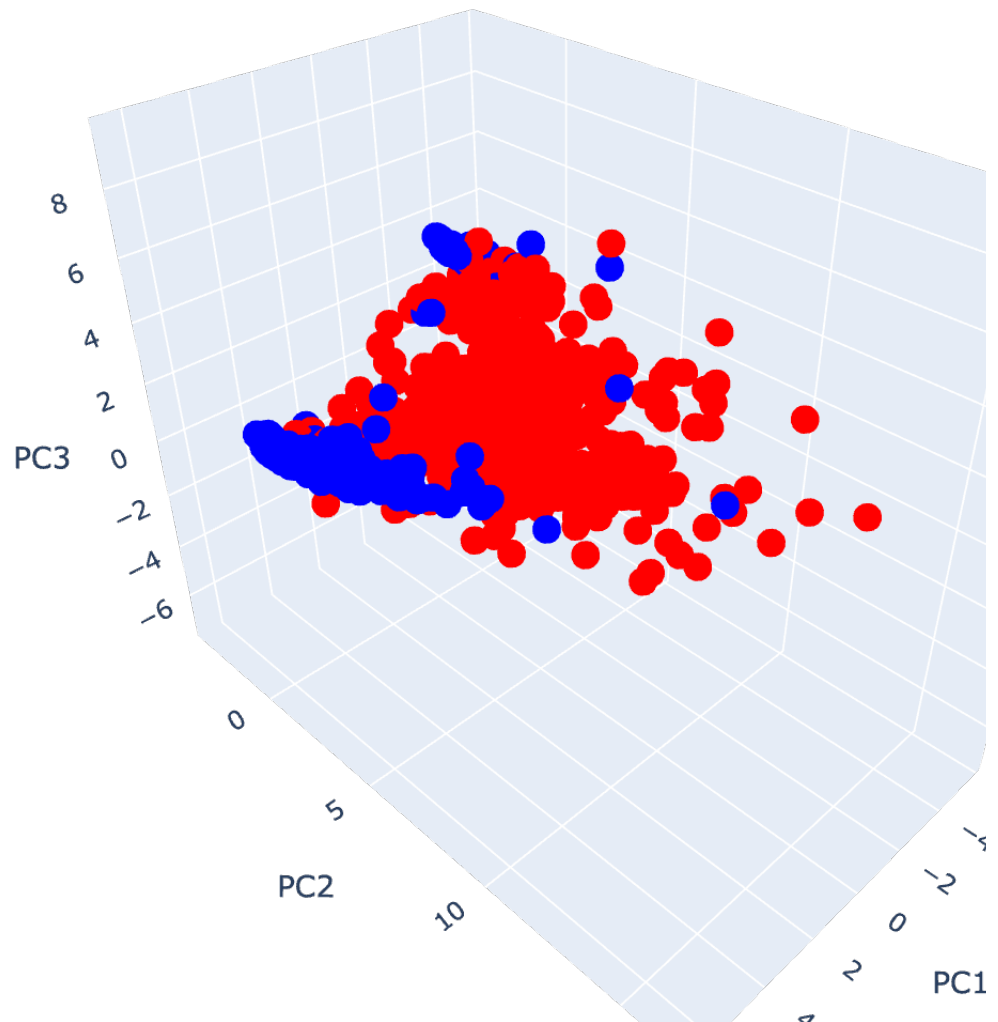
# 4. Reconstruction du DataFrame
df_pca = pd.DataFrame(X_pca, columns=["PC1", "PC2", "PC3"])
df_pca["Allergy_Present"] = y.values

# 5. Affichage graphique en 3D
fig = px.scatter_3d(
    df_pca,
    x="PC1",
    y="PC2",
    z="PC3",
    color=df_pca["Allergy_Present"].astype(str),
    title="Projection PCA en 3D des patients",
    labels={"color": "Présence d'allergie"},
    color_discrete_map={"0": "blue", "1": "red"},
    width=950,
    height=700
)

fig.update_layout(legend_title="Présence d'allergie")
fig.show()
```



## Projection PCA en 3D des patients



```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import pandas as pd
```

```
X= ALEX.copy()
X.drop(targets, axis=1, inplace=True)
X.drop(extra_columns, axis=1, inplace=True)
X.drop(extra, axis=1, inplace=True)
```

```
X = X.iloc[:, 1:]

y = ALEX["Respiratory_Allergy"]

# 3. Standardisation
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 4. PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Reconstruction d'un DataFrame avec les résultats PCA
df_pca = pd.DataFrame(X_pca, columns=["PC1", "PC2"])
df_pca["Respiratory_Allergy"] = y.values # Ajouter la cible

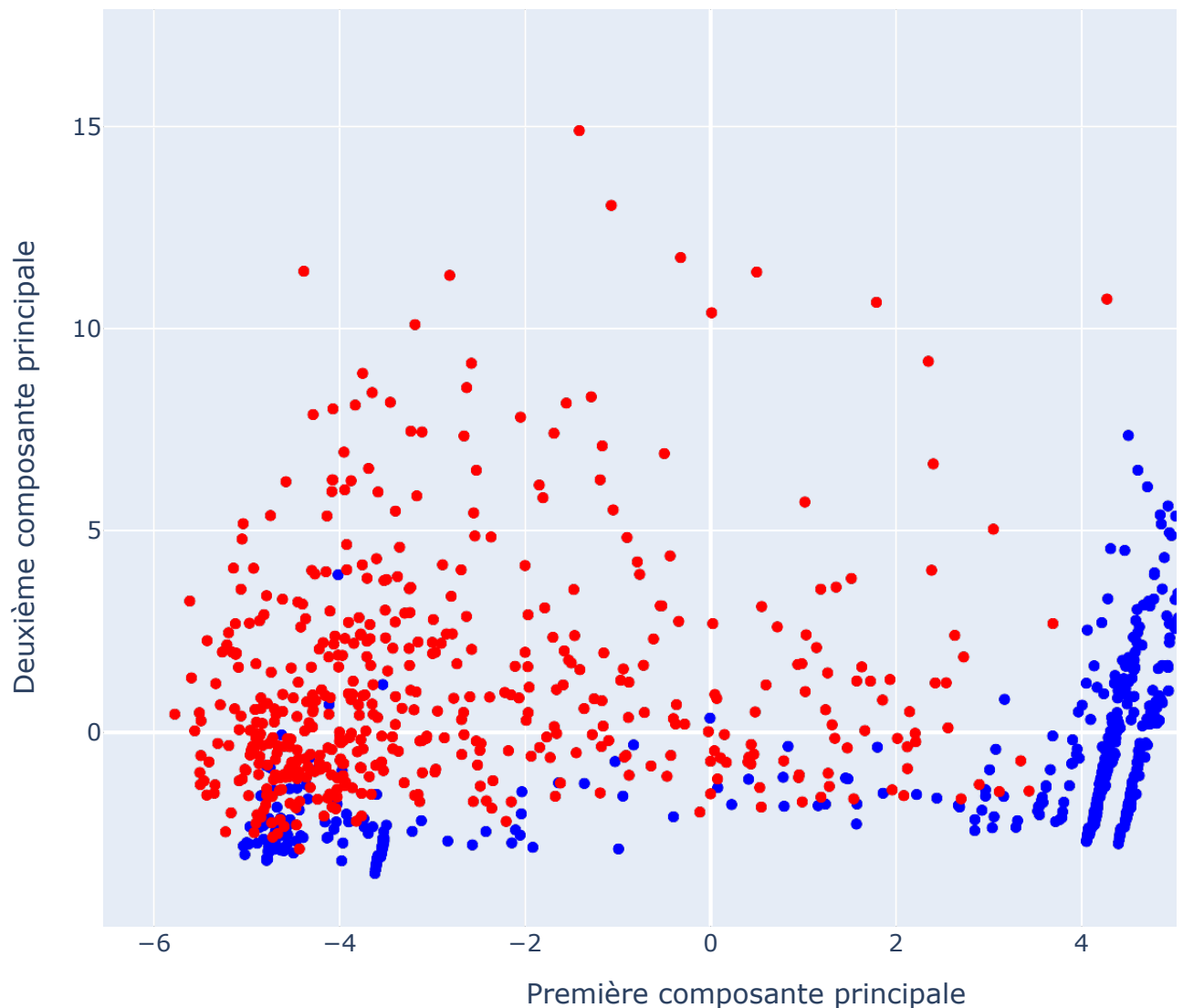
# Tracé du graphique
fig = px.scatter(
    df_pca,
    x="PC1",
    y="PC2",
    color=df_pca["Respiratory_Allergy"].astype(str), # couleur par classe 0 ou 1
    title="Projection PCA des patients",
    labels={
        "PC1": "Première composante principale",
        "PC2": "Deuxième composante principale",
        "color": "Présence d'allergie"
    },
    color_discrete_map={"0": "blue", "1": "red"},
    width=900,
    height=650
)

fig.update_layout(
    legend_title="Présence d'allergie"
)

fig.show()
```



## Projection PCA des patients



```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import pandas as pd
```

```
ALEX_aller = ALEX[ALEX["Allergy_Present"] == 1]
X= ALEX_aller.copy()
X.drop(targets, axis=1, inplace=True)
X.drop(extra_columns, axis=1, inplace=True)
X.drop(extra, axis=1, inplace=True)
```

```
X = X.iloc[:, 1:]
```



```
y = ALEX_aller["Severe_Allergy"]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

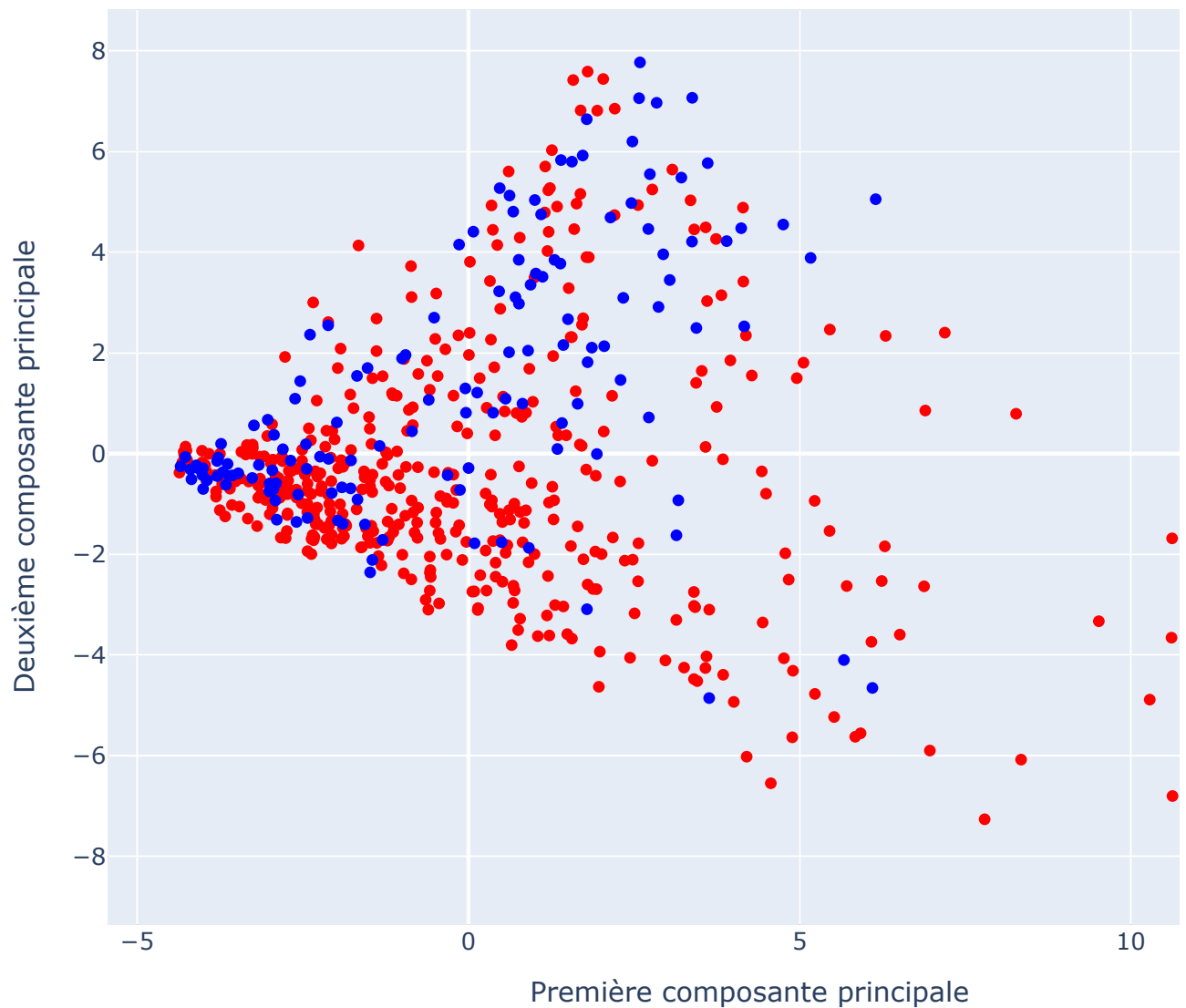
df_pca = pd.DataFrame(X_pca, columns=["PC1", "PC2"])
df_pca["Severe_Allergy"] = y.values # Ajouter la cible

fig = px.scatter(
    df_pca,
    x="PC1",
    y="PC2",
    color=df_pca["Severe_Allergy"].astype(str), # couleur par classe 0 ou 1
    title="Projection PCA des patients",
    labels={
        "PC1": "Première composante principale",
        "PC2": "Deuxième composante principale",
        "color": "Severe d'allergie"
    },
    color_discrete_map={"0": "blue", "1": "red"},
    width=900,
    height=650
)

fig.update_layout(
    legend_title="Présence d'allergie"
)
fig.show()
```



## Projection PCA des patients



```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import pandas as pd
```

```
ALEX_aller = ALEX[ALEX["Allergy_Present"] == 1]
X= ALEX_aller.copy()
X.drop(targets, axis=1, inplace=True)
X.drop(extra_columns, axis=1, inplace=True)
X.drop(extra, axis=1, inplace=True)
```

```
X = X.iloc[:, 1:]
```

```
y = ALEX_aller["Food_Allergy"]

# 3. Standardisation
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 4. PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Reconstruction d'un DataFrame avec les résultats PCA
df_pca = pd.DataFrame(X_pca, columns=["PC1", "PC2"])
df_pca["Food_Allergy"] = y.values # Ajouter la cible

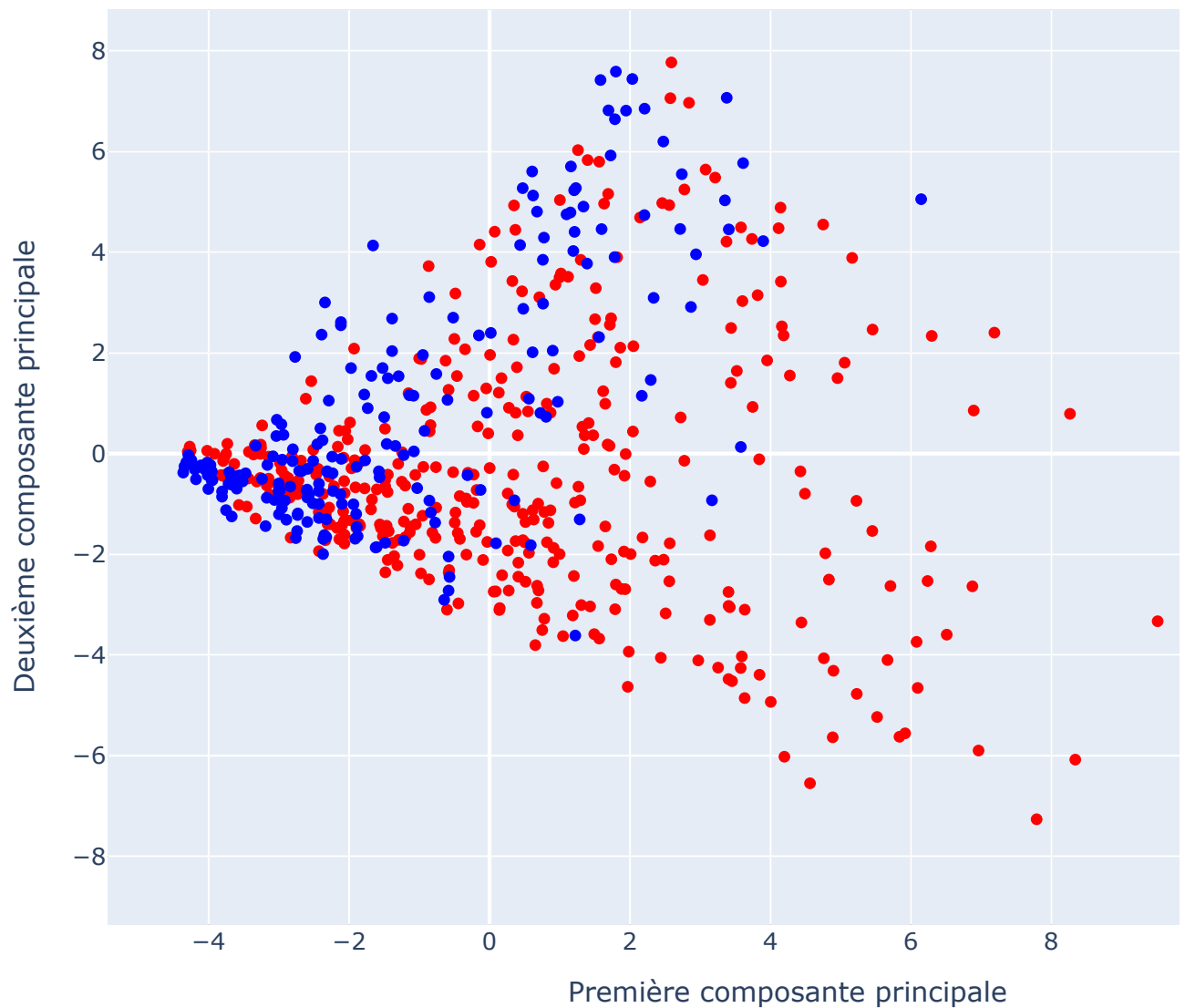
# Tracé du graphique
fig = px.scatter(
    df_pca,
    x="PC1",
    y="PC2",
    color=df_pca["Food_Allergy"].astype(str), # couleur par classe 0 ou 1
    title="Projection PCA des patients",
    labels={
        "PC1": "Première composante principale",
        "PC2": "Deuxième composante principale",
        "color": "Food d'allergie"
    },
    color_discrete_map={"0": "blue", "1": "red"},
    width=900,
    height=650
)

fig.update_layout(
    legend_title="Food Allergie"
)

fig.show()
```



## Projection PCA des patients



```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import pandas as pd
```

```
ALEX_aller = ALEX[ALEX["Allergy_Present"] == 1]
X= ALEX_aller.copy()
X.drop(targets, axis=1, inplace=True)
X.drop(extra_columns, axis=1, inplace=True)
X.drop(extra, axis=1, inplace=True)
```

```
X = X.iloc[:, 1:]
```

```
y = ALEX_aller["Venom_Allergy"]

# 3. Standardisation
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 4. PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Reconstruction d'un DataFrame avec les résultats PCA
df_pca = pd.DataFrame(X_pca, columns=["PC1", "PC2"])
df_pca["Venom_Allergy"] = y.values # Ajouter la cible

# Tracé du graphique
fig = px.scatter(
    df_pca,
    x="PC1",
    y="PC2",
    color=df_pca["Venom_Allergy"].astype(str), # couleur par classe 0 ou 1
    title="Projection PCA des patients",
    labels={
        "PC1": "Première composante principale",
        "PC2": "Deuxième composante principale",
        "color": "Venom d'allergie"
    },
    color_discrete_map={"0": "blue", "1": "red"},
    width=900,
    height=650
)

fig.update_layout(
    legend_title="Venom Allergie"
)

fig.show()
```



## Projection PCA des patients

