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"
V2_df = df[df["Chip_Type"] == "ISAC_V2"]
V2_df.head()
```

	Unnamed: 0	Chip_Code	Chip_Type	Chip_Image_Nam
1121	WQW0008	202272574	ISAC_V2	DYF0F30_4_202272574_2023_1_3_16_52_6.bm
1122	WQW0054	210553086	ISAC_V2	E4J0C30_1_210553086_2023_1_3_16_49_12.bm
1123	WQW0059	210993011	ISAC_V2	Na
1124	WQW0160	221402048	ISAC_V2	EK41V30_4_221402048_2023_1_3_16_27_36.bm
1125	WQW0189	223042355	ISAC_V2	END0D30_4_223042355_2023_1_3_16_20_18.bm

5 rows × 386 columns

```
V2_df.describe()
```

	Age	Blood_Month_sample	French_Residence_Department	French_Re
count	720.000000	720.000000	781.000000	781.00
mean	21.680556	6.447222	254.354673	6.02
std	18.906642	3.139323	387.690434	2.43
min	0.000000	1.000000	2.000000	1.00
25%	8.000000	4.000000	51.000000	6.00
50%	14.000000	7.000000	62.000000	7.00
75%	32.000000	9.000000	92.000000	8.00
max	83.000000	12.000000	999.000000	11.00

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.

```
V2 = V2_df.dropna(axis=1)
V2.describe()
```

	French_Residence_Department	French_Region	Rural_or_urban_area	Alle
count	781.000000	781.000000	781.000000	
mean	254.354673	6.024328	4.149808	
std	387.690434	2.437563	4.089930	
min	2.000000	1.000000	0.000000	
25%	51.000000	6.000000	1.000000	
50%	62.000000	7.000000	1.000000	
75%	92.000000	8.000000	9.000000	
max	999.000000	11.000000	9.000000	

8 rows × 158 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]
}

V2 = expand_custom_one_hot(V2, value_lists)
V2 = V2.drop('Type_of_Respiratory_Allergy', axis=1)
V2 = V2.drop('Type_of_Food_Allergy', axis=1)
V2 = V2.drop('Type_of_Venom_Allergy', axis=1)
V2 = V2.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

```
V2.to_excel('V2.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_sympto
    '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_sympto
    '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
allergenes = {
    "Pollens": [
        "Aln_g_1", "Amb_a_1", "Art_v_1", "Art_v_3", "Bet_v_1", "Bet_v_2", "Bet_
        "Che_a_1", "Cry_j_1", "Cup_a_1", "Cyn_d_1", "Mer_a_1", "Ole_e_1", "Ole_
        "Ole_e_9", "Par_j_2", "Phl_p_1", "Phl_p_2", "Phl_p_4", "Phl_p_5", "Phl_
        "Phl_p_7", "Phl_p_11", "Phl_p_12", "Pla_a_1", "Pla_a_3", "Pla_l_1", "Sa
    ],
    "Moisissures": [
        "Alt_a_1", "Alt_a_6", "Asp_f_1", "Asp_f_3", "Asp_f_6", "Cla_h_8",
        "Pen_m_1", "Pen_m_2", "Pen_m_4"
    ],
    "Animaux": [
        "Bla_g_1", "Bla_g_2", "Bla_g_5", "Bla_g_7", "Blo_t_5", "Can_f_1", "Can_
        "Can_f_3", "Can_f_4", "Can_f_5", "Can_f_6", "Equ_c_1", "Equ_c_3", "Fel_
        "Fel_d_2", "Fel_d_4", "Mus_m_1", "Der_f_1", "Der_f_2", "Der_p_1", "Der_
        "Der_p_10", "Der_p_23", "Lep_d_2", "Gad_c_1", "Bos_d_4", "Bos_d_5", "Bo
        "Bos_d_8", "Bos_d_Lactoferrin"
    ],
    "Alimentaires": [
        "Act_d_1", "Act_d_2", "Act_d_5", "Act_d_8", "Ana_o_2", "Ani_s_1", "Ani_
        "Api_g_1", "Ara_h_1", "Ara_h_2", "Ara_h_3", "Ara_h_6", "Ara_h_8", "Ara_
        "Ber_e_1", "Cor_a_1.0101", "Cor_a_1.0401", "Cor_a_8", "Cor_a_9", "Cor_a
        "Fag_e_2", "Gal_d_1", "Gal_d_2", "Gal_d_3", "Gal_d_5", "Gly_m_4", "Gly_
        "Gly_m_6", "Jug_r_1", "Jug_r_3", "Mal_d_1", "Pru_p_1", "Pru_p_3", "Sal_
        "Ses_i_1", "Tri_a_14", "Tri_a_19.0101", "Tri_a_aA_TI", "Alpha-Gal", "Ar
    ],
    "Glycanes": [
        "Hev_b_1", "Hev_b_3", "Hev_b_5", "Hev_b_6", "Hev_b_8", "MUXF3"
    ],
    "Autres": [
        "Che_a_1", "Mer_a_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 V2.columns]
    V2[f"Moyenne_{type_allergene}"] = V2[colonnes_presentes].mean(axis=1)
    V2[f"Max_{type_allergene}"] = V2[colonnes_presentes].max(axis=1)
    V2[f"Min_{type_allergene}"] = V2[colonnes_presentes].min(axis=1)
```

```
V2.describe()
```

	French_Residence_Department	French_Region	Rural_or_urban_area	Alle
count	781.000000	781.000000	781.000000	
mean	254.354673	6.024328	4.149808	
std	387.690434	2.437563	4.089930	
min	2.000000	1.000000	0.000000	
25%	51.000000	6.000000	1.000000	
50%	62.000000	7.000000	1.000000	
75%	92.000000	8.000000	9.000000	
max	999.000000	11.000000	9.000000	

8 rows × 233 columns

```
col_allergenes = [

        "Aln_g_1", "Amb_a_1", "Art_v_1", "Art_v_3", "Bet_v_1", "Bet_v_2", "Bet_
        "Che_a_1", "Cry_j_1", "Cup_a_1", "Cyn_d_1", "Mer_a_1", "Ole_e_1", "Ole_
        "Ole_e_9", "Par_j_2", "Phl_p_1", "Phl_p_2", "Phl_p_4", "Phl_p_5", "Phl_
        "Phl_p_7", "Phl_p_11", "Phl_p_12", "Pla_a_1", "Pla_a_3", "Pla_l_1", "Sa

        "Alt_a_1", "Alt_a_6", "Asp_f_1", "Asp_f_3", "Asp_f_6", "Cla_h_8",
        "Pen_m_1", "Pen_m_2", "Pen_m_4",
        "Bla_g_1", "Bla_g_2", "Bla_g_5", "Bla_g_7", "Blo_t_5", "Can_f_1", "Can_
        "Can_f_3", "Can_f_4", "Can_f_5", "Can_f_6", "Equ_c_1", "Equ_c_3", "Fel_
        "Fel_d_2", "Fel_d_4", "Mus_m_1", "Der_f_1", "Der_f_2", "Der_p_1", "Der_
        "Der_p_10", "Der_p_23", "Lep_d_2", "Gad_c_1", "Bos_d_4", "Bos_d_5", "Bo
        "Bos_d_8", "Bos_d_Lactoferrin",

        "Act_d_1", "Act_d_2", "Act_d_5", "Act_d_8", "Ana_o_2", "Ani_s_1", "Ani_
        "Api_g_1", "Ara_h_1", "Ara_h_2", "Ara_h_3", "Ara_h_6", "Ara_h_8", "Ara_
        "Ber_e_1", "Cor_a_1.0101", "Cor_a_1.0401", "Cor_a_8", "Cor_a_9", "Cor_a
        "Fag_e_2", "Gal_d_1", "Gal_d_2", "Gal_d_3", "Gal_d_5", "Gly_m_4", "Gly_
        "Gly_m_6", "Jug_r_1", "Jug_r_3", "Mal_d_1", "Pru_p_1", "Pru_p_3", "Sal_
        "Ses_i_1", "Tri_a_14", "Tri_a_19.0101", "Tri_a_aA_TI", "Alpha-Gal", "Ar
        "Hev_b_1", "Hev_b_3", "Hev_b_5", "Hev_b_6", "Hev_b_8", "MUXF3",
        "Che_a_1", "Mer_a_1",
    ]
V2.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

```
V2.loc[V2["Allergy_Present"] == 0, ["Respiratory_Allergy", "Food_Allergy", "Ver
V2.loc[V2["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

V2.loc[V2["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_TPO",
    "Type_of_Food_Allergy_Tree_Nuts"
]] = 0

V2.loc[V2["Venom_Allergy"] == 0, ["Type_of_Venom_Allergy_ATCD_Venom",
    "Type_of_Venom_Allergy_IGE_Venom"]] = 0
```

```
V2.describe()
```

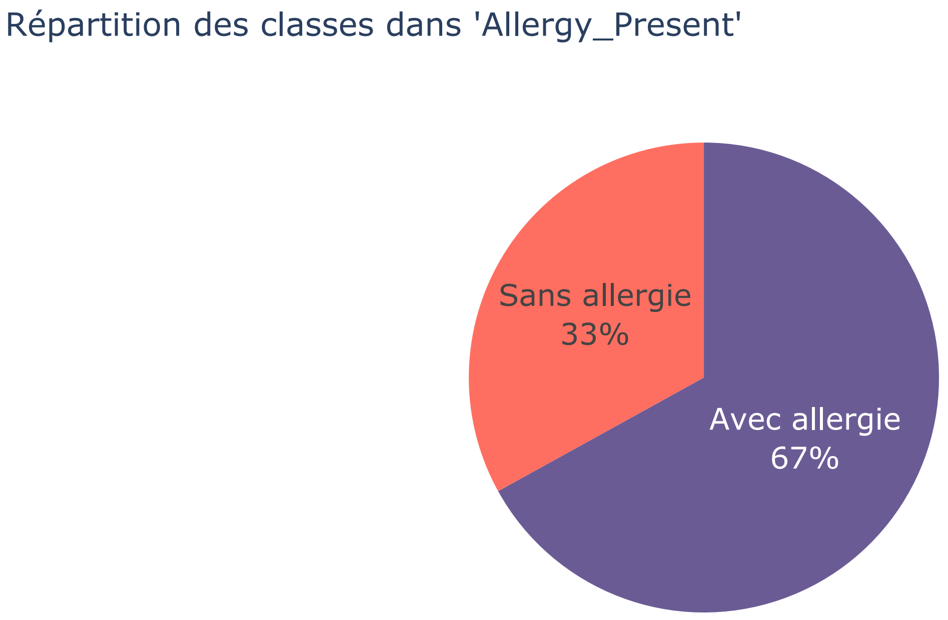
	French_Residence_Department	French_Region	Rural_or_urban_area	Alle
count	781.000000	781.000000	781.000000	
mean	254.354673	6.024328	4.149808	
std	387.690434	2.437563	4.089930	
min	2.000000	1.000000	0.000000	
25%	51.000000	6.000000	1.000000	
50%	62.000000	7.000000	1.000000	
75%	92.000000	8.000000	9.000000	
max	999.000000	11.000000	9.000000	

8 rows × 121 columns

```
V2.to_excel('ISAC_V2.xlsx', index=False)
```

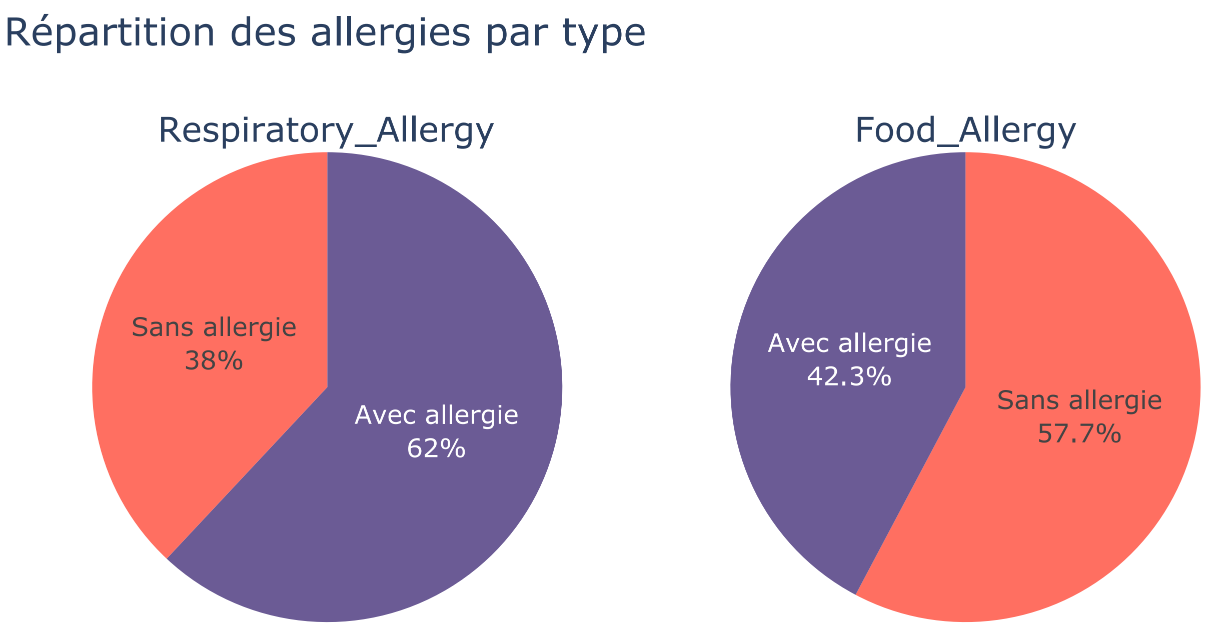
Repartition suivant chaque Target

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

value_counts = V2["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()
```



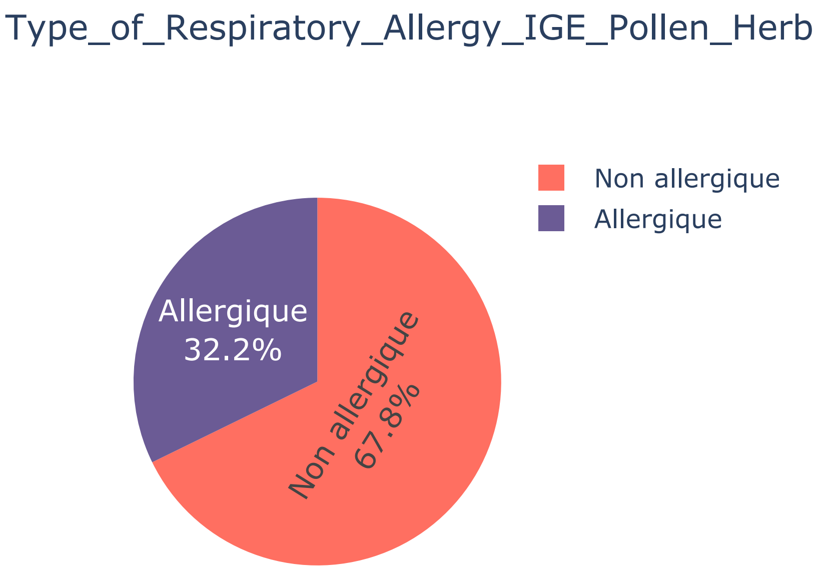
```
import pandas as pd
import plotly.graph_objects as go
from plotly.subplots import make_subplots

allergy_positive = V2[V2["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"
}
```

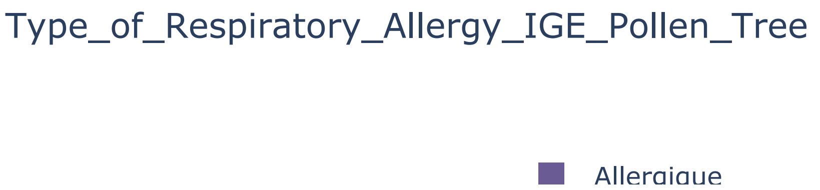
```
fig = make_subplots(rows=1, cols=3, specs=[[{'type':'domain'}]*3],
                    subplot_titles=targets)

for i, col in enumerate(targets):
    counts = V2[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()
```

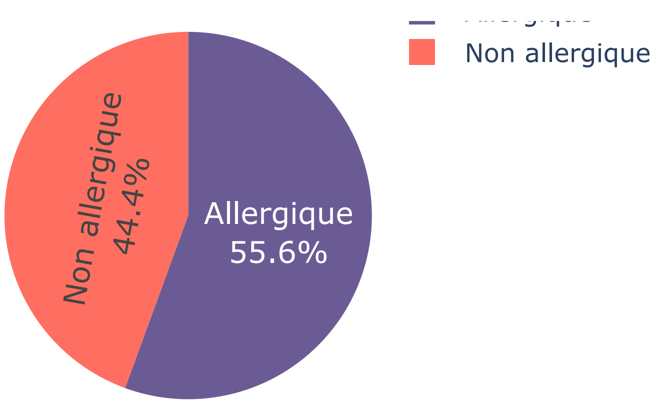


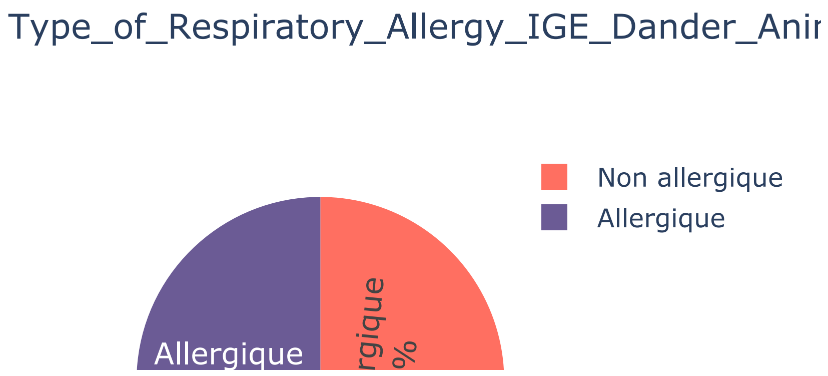
```
import pandas as pd
import plotly.express as px

res_pos = V2[V2["Respiratory_Allergy"] == 1]
food_pos = V2[V2["Food_Allergy"] == 1]
venom_pos = V2[V2["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()
```

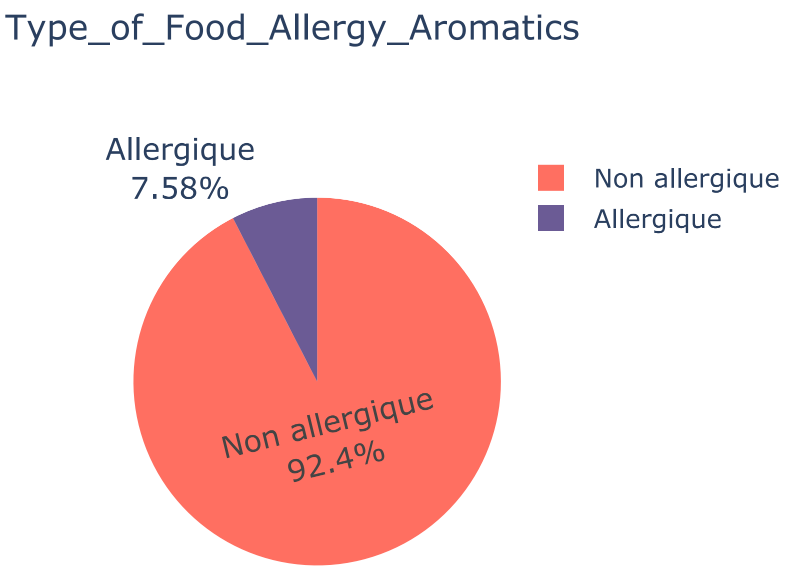


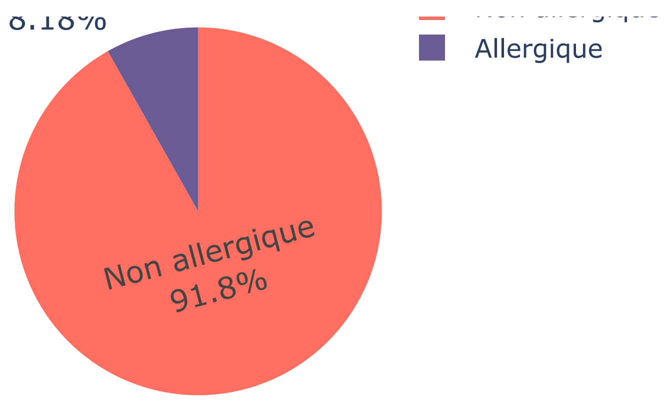




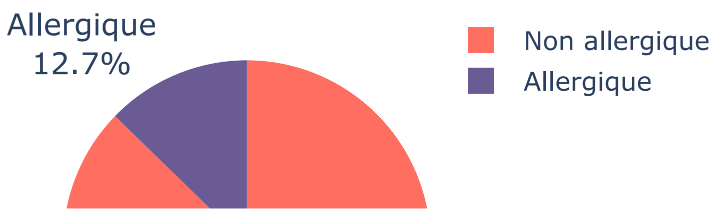


```
food_subtypes = [
    "Type_of_Food_Allergy_Aromatics",
    "Type_of_Food_Allergy_Cereals_&_Seeds",
    "Type_of_Food_Allergy_Egg",
    "Type_of_Food_Allergy_Fish",
    "Type_of_Food_Allergy_Fruits_and_Vegetables",
    "Type_of_Food_Allergy_Mammalian_Milk",
    "Type_of_Food_Allergy_Oral_Syndrom",
    "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()
```



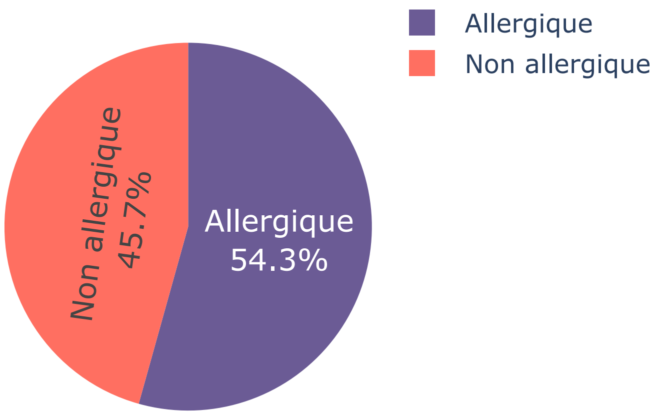


Type_of_Food_Allergy_Egg



Type_of_Respiratory_Allergy_IGE_Pollen_Gram

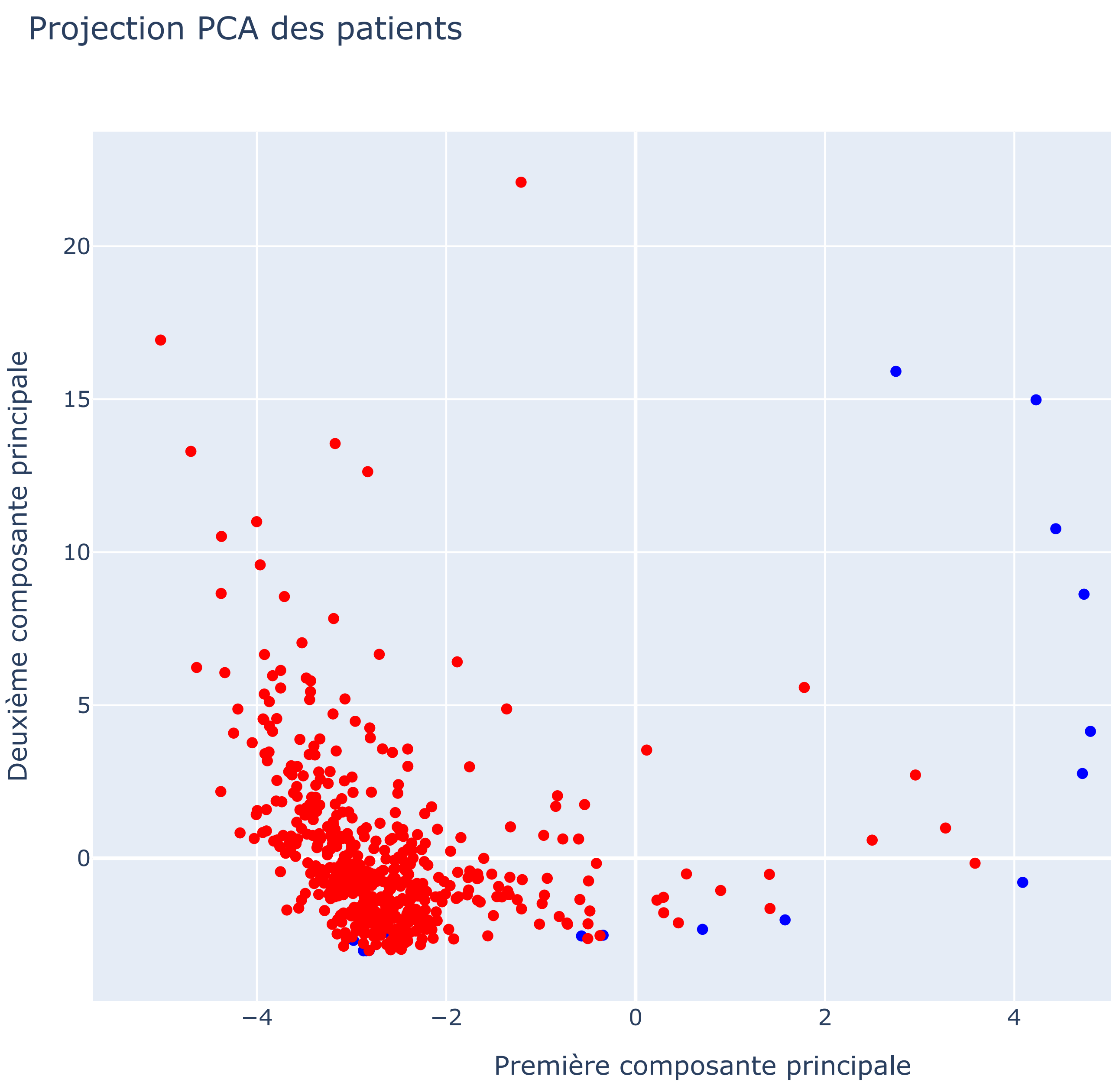
PCA et UMAP



Type_of_Respiratory_Allergy_GINA

```
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",
    ]

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= V2.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:]
```

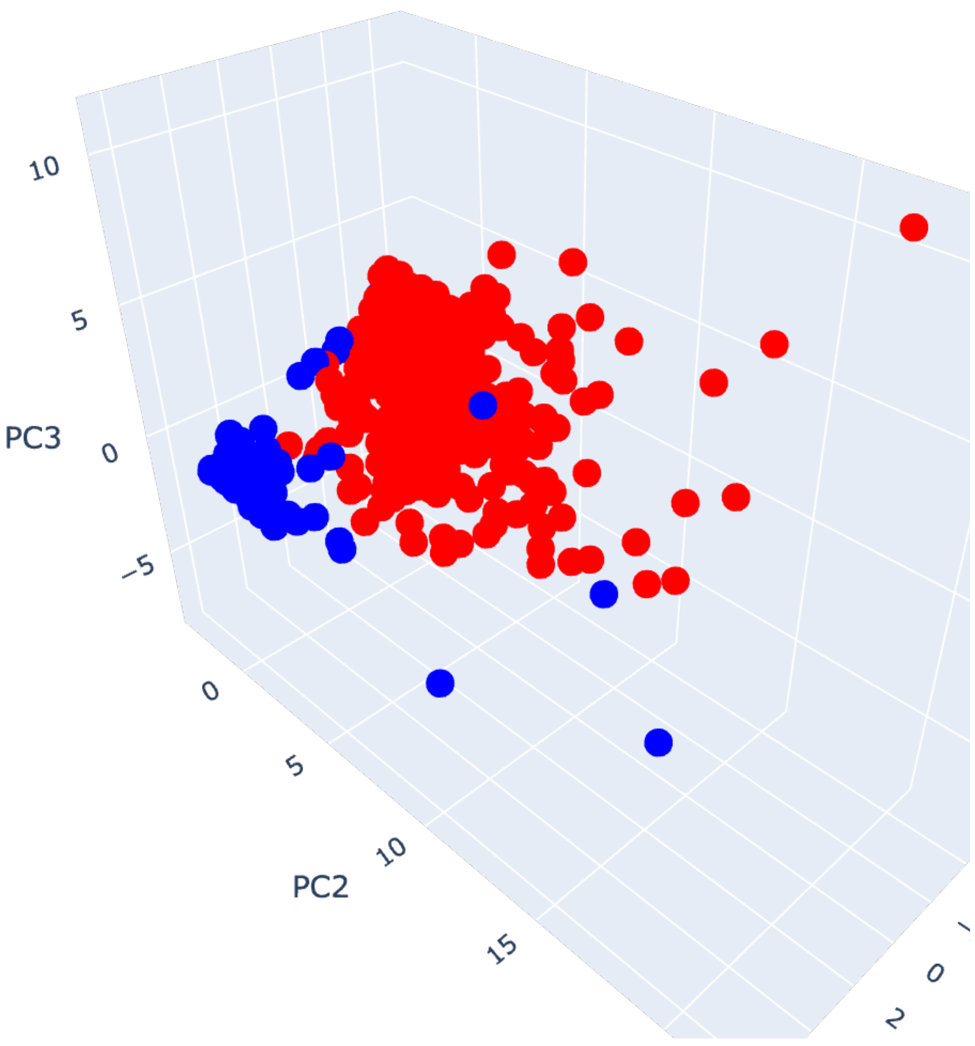
```
y = V2["Allergy_Present"]


# 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()
```



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

# 1. Préparation des données
X = V2.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:]
```

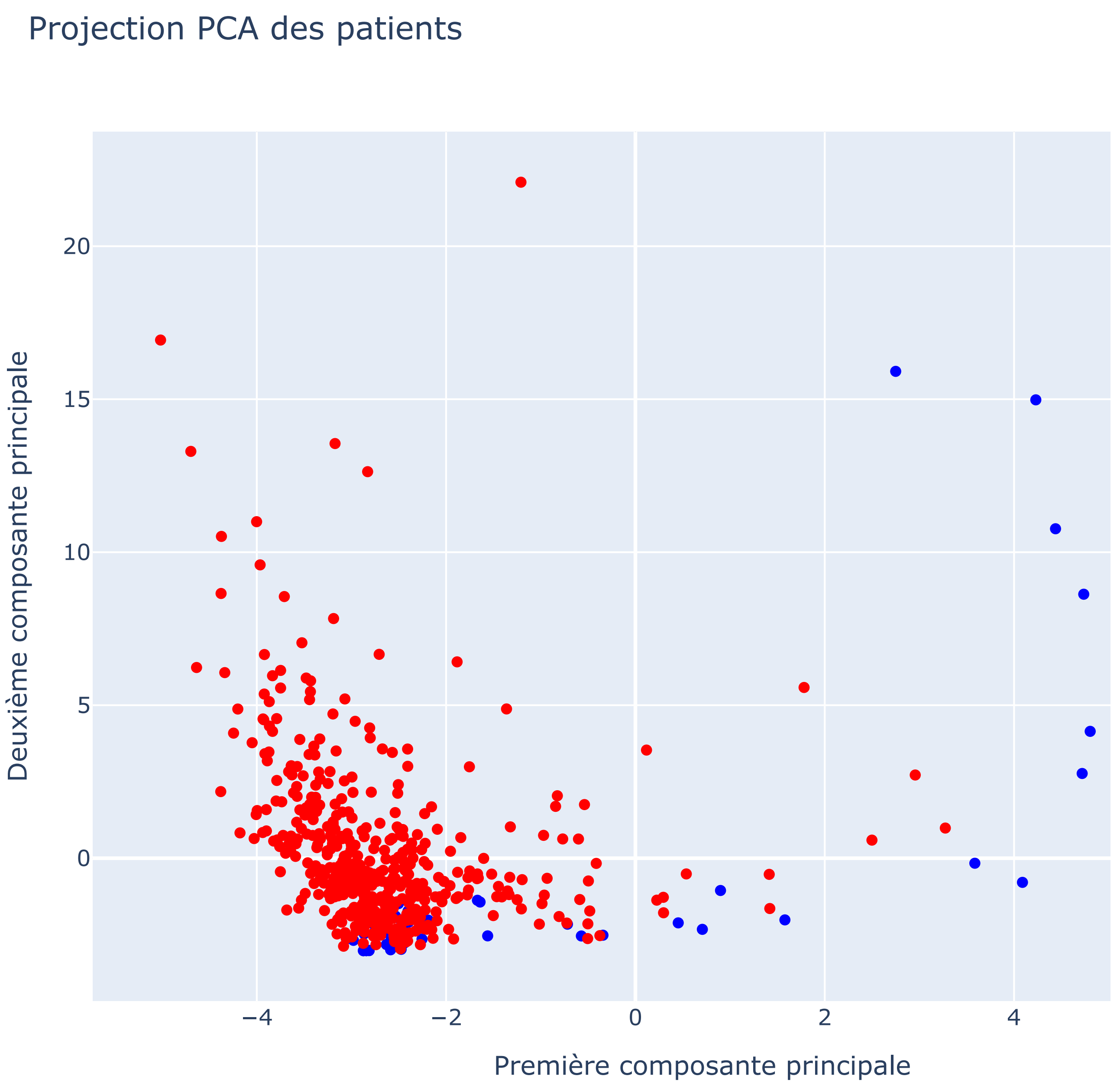
```
y = V2["Allergy_Present"]

# 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



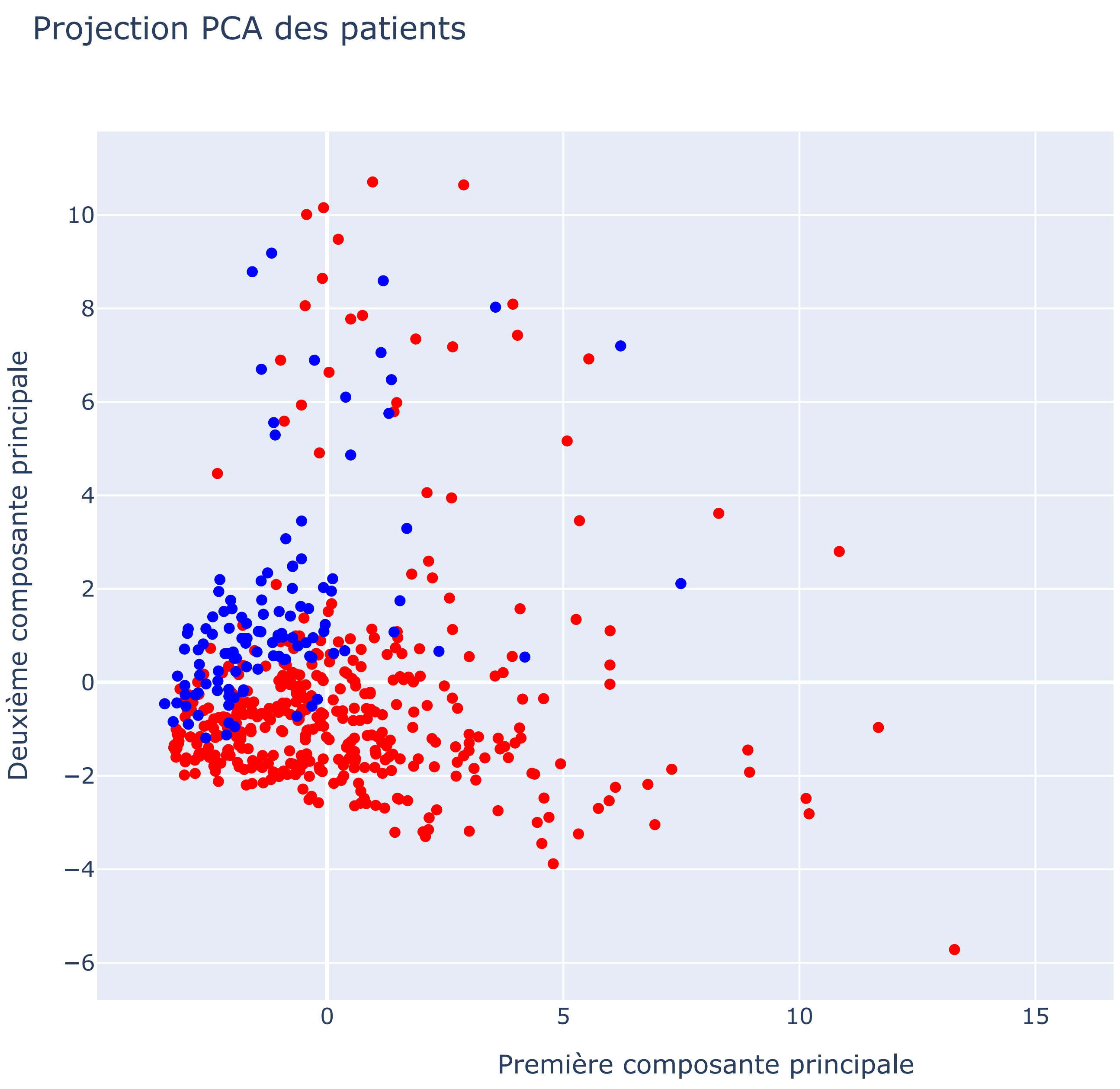
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import pandas as pd

X= V2.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 = V2["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
    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()
```



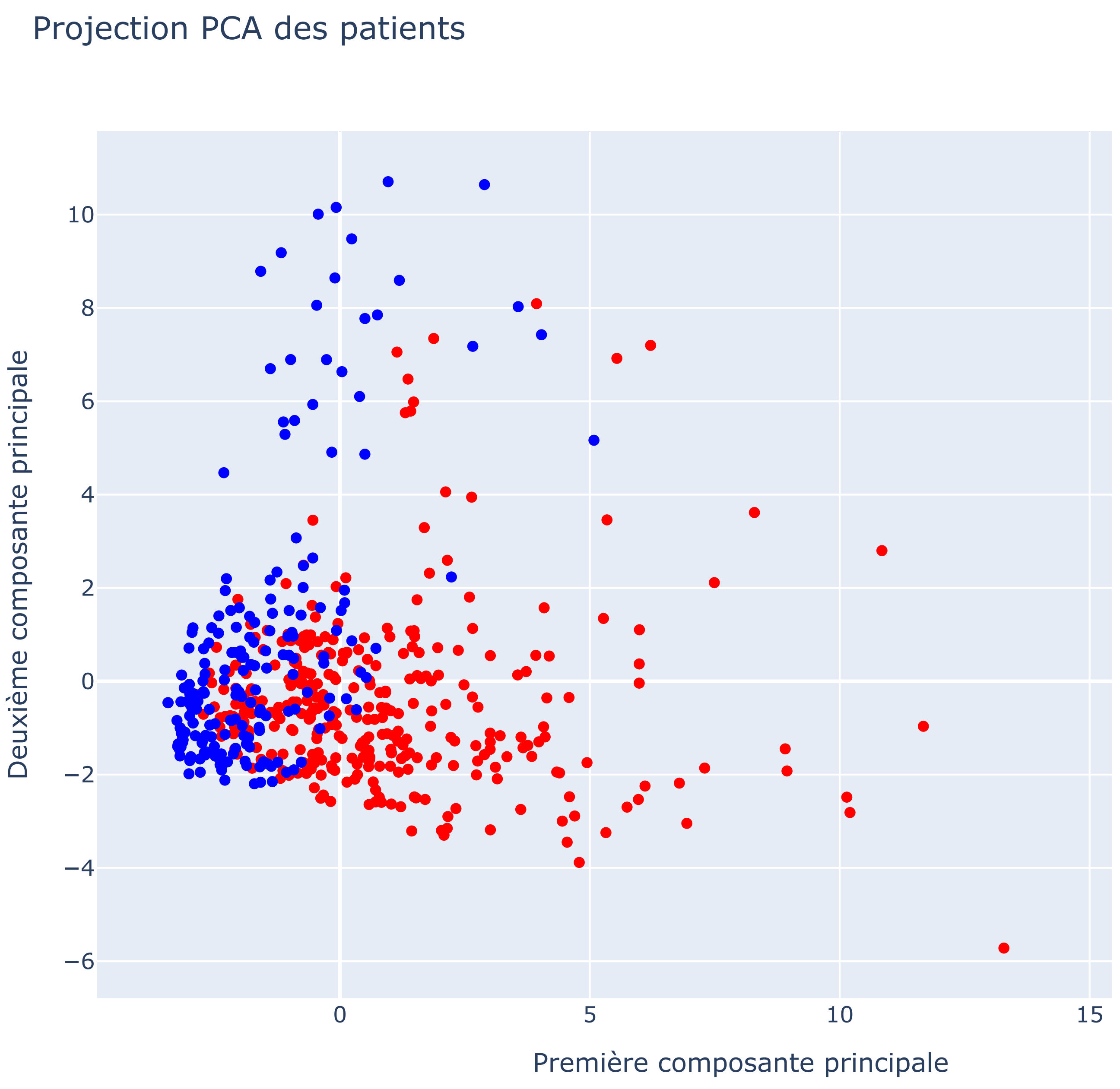
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import pandas as pd

V2_aller = V2[V2["Allergy_Present"] == 1]
X= V2_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 = V2_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()
```



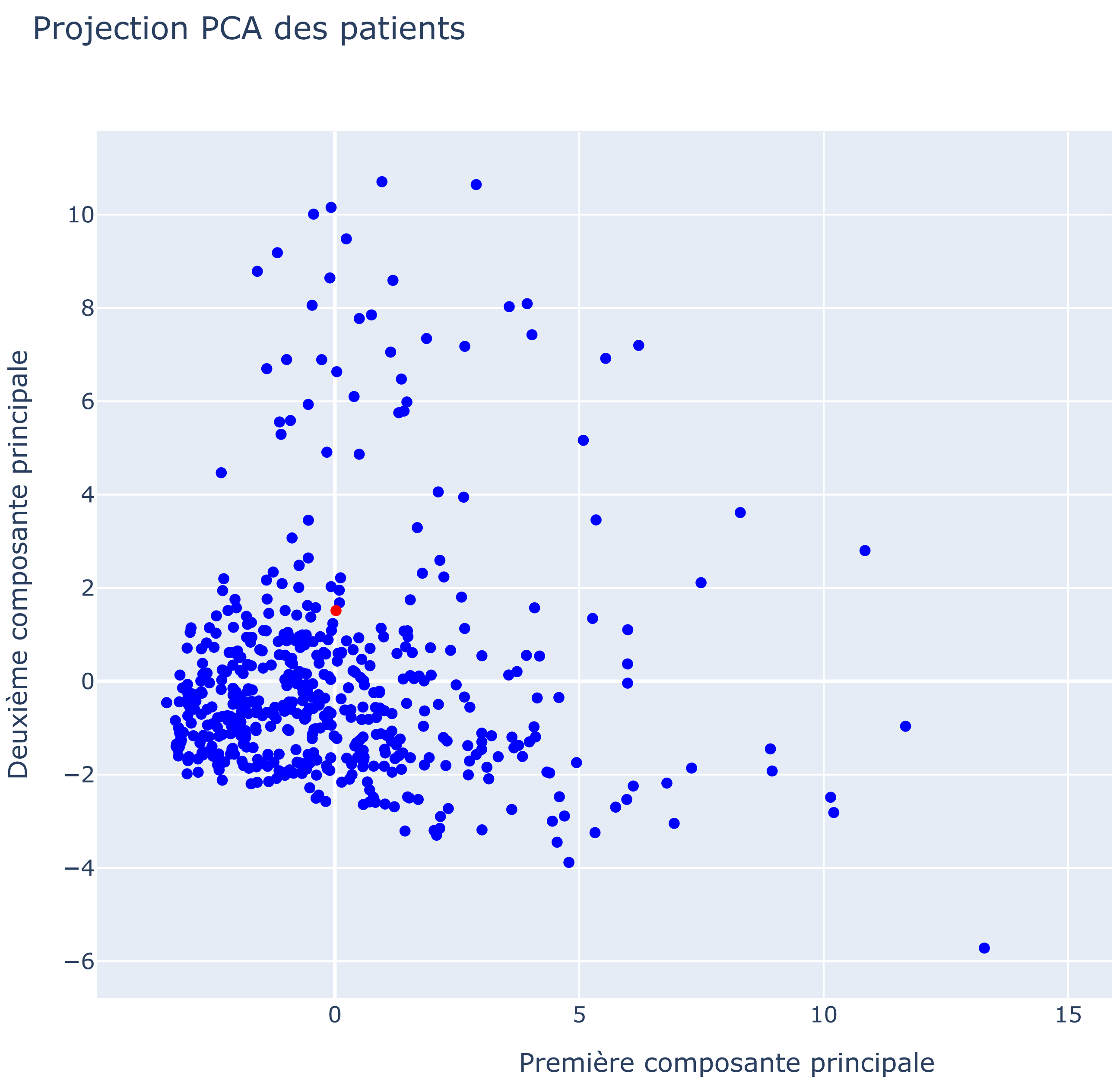
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import pandas as pd

V2_aller = V2[V2["Allergy_Present"] == 1]
X= V2_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 = V2_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()
```



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

V1_aller = V2[V2["Allergy_Present"] == 1]
X= V2_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 = V2_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
Deuxième composante principale
Première composante principale
10
8
6
4
2
0
−2
−4
−6
0
5
10
15