

> 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.brr
1122	WQW0054	210553086	ISAC_V2	E4J0C30_1_210553086_2023_1_3_16_49_12.brr
1123	WQW0059	210993011	ISAC_V2	Na
1124	WQW0160	221402048	ISAC_V2	EK41V30_4_221402048_2023_1_3_16_27_36.brr
1125	WQW0189	223042355	ISAC_V2	END0D30_4_223042355_2023_1_3_16_20_18.brr

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_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
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", "Bc
        "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)

```

	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	

```
col_allergenes = [
    "Aln_g_1", "Amb_a_1", "Art_v_1", "Art_v_3", "Bet_v_1", "Bet_v_2", "Bet_v_3",
    "Che_a_1", "Cry_j_1", "Cup_a_1", "Cyn_d_1", "Mer_a_1", "Ole_e_1", "Ole_e_2",
    "Ole_e_9", "Par_j_2", "Phl_p_1", "Phl_p_2", "Phl_p_4", "Phl_p_5", "Phl_p_6",
    "Phl_p_7", "Phl_p_11", "Phl_p_12", "Pla_a_1", "Pla_a_3", "Pla_l_1", "Sap_m_1",
    "Sap_m_2", "Sap_m_3", "Sap_m_4", "Sap_m_5", "Sap_m_6", "Sap_m_7", "Sap_m_8",
    "Sap_m_9", "Sap_m_10", "Sap_m_11", "Sap_m_12", "Sap_m_13", "Sap_m_14", "Sap_m_15",
    "Sap_m_16", "Sap_m_17", "Sap_m_18", "Sap_m_19", "Sap_m_20", "Sap_m_21", "Sap_m_22",
    "Sap_m_23", "Sap_m_24", "Sap_m_25", "Sap_m_26", "Sap_m_27", "Sap_m_28", "Sap_m_29",
    "Sap_m_30", "Sap_m_31", "Sap_m_32", "Sap_m_33", "Sap_m_34", "Sap_m_35", "Sap_m_36",
    "Sap_m_37", "Sap_m_38", "Sap_m_39", "Sap_m_40", "Sap_m_41", "Sap_m_42", "Sap_m_43",
    "Sap_m_44", "Sap_m_45", "Sap_m_46", "Sap_m_47", "Sap_m_48", "Sap_m_49", "Sap_m_50",
    "Sap_m_51", "Sap_m_52", "Sap_m_53", "Sap_m_54", "Sap_m_55", "Sap_m_56", "Sap_m_57",
    "Sap_m_58", "Sap_m_59", "Sap_m_60", "Sap_m_61", "Sap_m_62", "Sap_m_63", "Sap_m_64",
    "Sap_m_65", "Sap_m_66", "Sap_m_67", "Sap_m_68", "Sap_m_69", "Sap_m_70", "Sap_m_71",
    "Sap_m_72", "Sap_m_73", "Sap_m_74", "Sap_m_75", "Sap_m_76", "Sap_m_77", "Sap_m_78",
    "Sap_m_79", "Sap_m_80", "Sap_m_81", "Sap_m_82", "Sap_m_83", "Sap_m_84", "Sap_m_85",
    "Sap_m_86", "Sap_m_87", "Sap_m_88", "Sap_m_89", "Sap_m_90", "Sap_m_91", "Sap_m_92",
    "Sap_m_93", "Sap_m_94", "Sap_m_95", "Sap_m_96", "Sap_m_97", "Sap_m_98", "Sap_m_99",
    "Sap_m_100", "Sap_m_101", "Sap_m_102", "Sap_m_103", "Sap_m_104", "Sap_m_105",
    "Sap_m_106", "Sap_m_107", "Sap_m_108", "Sap_m_109", "Sap_m_110", "Sap_m_111",
    "Sap_m_112", "Sap_m_113", "Sap_m_114", "Sap_m_115", "Sap_m_116", "Sap_m_117",
    "Sap_m_118", "Sap_m_119", "Sap_m_120", "Sap_m_121", "Sap_m_122", "Sap_m_123",
    "Sap_m_124", "Sap_m_125", "Sap_m_126", "Sap_m_127", "Sap_m_128", "Sap_m_129",
    "Sap_m_130", "Sap_m_131", "Sap_m_132", "Sap_m_133", "Sap_m_134", "Sap_m_135",
    "Sap_m_136", "Sap_m_137", "Sap_m_138", "Sap_m_139", "Sap_m_140", "Sap_m_141",
    "Sap_m_142", "Sap_m_143", "Sap_m_144", "Sap_m_145", "Sap_m_146", "Sap_m_147",
    "Sap_m_148", "Sap_m_149", "Sap_m_150", "Sap_m_151", "Sap_m_152", "Sap_m_153",
    "Sap_m_154", "Sap_m_155", "Sap_m_156", "Sap_m_157", "Sap_m_158", "Sap_m_159",
    "Sap_m_160", "Sap_m_161", "Sap_m_162", "Sap_m_163", "Sap_m_164", "Sap_m_165",
    "Sap_m_166", "Sap_m_167", "Sap_m_168", "Sap_m_169", "Sap_m_170", "Sap_m_171",
    "Sap_m_172", "Sap_m_173", "Sap_m_174", "Sap_m_175", "Sap_m_176", "Sap_m_177",
    "Sap_m_178", "Sap_m_179", "Sap_m_180", "Sap_m_181", "Sap_m_182", "Sap_m_183",
    "Sap_m_184", "Sap_m_185", "Sap_m_186", "Sap_m_187", "Sap_m_188", "Sap_m_189",
    "Sap_m_190", "Sap_m_191", "Sap_m_192", "Sap_m_193", "Sap_m_194", "Sap_m_195",
    "Sap_m_196", "Sap_m_197", "Sap_m_198", "Sap_m_199", "Sap_m_200", "Sap_m_201",
    "Sap_m_202", "Sap_m_203", "Sap_m_204", "Sap_m_205", "Sap_m_206", "Sap_m_207",
    "Sap_m_208", "Sap_m_209", "Sap_m_210", "Sap_m_211", "Sap_m_212", "Sap_m_213",
    "Sap_m_214", "Sap_m_215", "Sap_m_216", "Sap_m_217", "Sap_m_218", "Sap_m_219",
    "Sap_m_220", "Sap_m_221", "Sap_m_222", "Sap_m_223", "Sap_m_224", "Sap_m_225",
    "Sap_m_226", "Sap_m_227", "Sap_m_228", "Sap_m_229", "Sap_m_230", "Sap_m_231",
    "Sap_m_232", "Sap_m_233", "Sap_m_234", "Sap_m_235", "Sap_m_236", "Sap_m_237",
    "Sap_m_238", "Sap_m_239", "Sap_m_240", "Sap_m_241", "Sap_m_242", "Sap_m_243",
    "Sap_m_244", "Sap_m_245", "Sap_m_246", "Sap_m_247", "Sap_m_248", "Sap_m_249",
    "Sap_m_250", "Sap_m_251", "Sap_m_252", "Sap_m_253", "Sap_m_254", "Sap_m_255",
    "Sap_m_256", "Sap_m_257", "Sap_m_258", "Sap_m_259", "Sap_m_260", "Sap_m_261",
    "Sap_m_262", "Sap_m_263", "Sap_m_264", "Sap_m_265", "Sap_m_266", "Sap_m_267",
    "Sap_m_268", "Sap_m_269", "Sap_m_270", "Sap_m_271", "Sap_m_272", "Sap_m_273",
    "Sap_m_274", "Sap_m_275", "Sap_m_276", "Sap_m_277", "Sap_m_278", "Sap_m_279",
    "Sap_m_280", "Sap_m_281", "Sap_m_282", "Sap_m_283", "Sap_m_284", "Sap_m_285",
    "Sap_m_286", "Sap_m_287", "Sap_m_288", "Sap_m_289", "Sap_m_290", "Sap_m_291",
    "Sap_m_292", "Sap_m_293", "Sap_m_294", "Sap_m_295", "Sap_m_296", "Sap_m_297",
    "Sap_m_298", "Sap_m_299", "Sap_m_300", "Sap_m_301", "Sap_m_302", "Sap_m_303",
    "Sap_m_304", "Sap_m_305", "Sap_m_306", "Sap_m_307", "Sap_m_308", "Sap_m_309",
    "Sap_m_310", "Sap_m_311", "Sap_m_312", "Sap_m_313", "Sap_m_314", "Sap_m_315",
    "Sap_m_316", "Sap_m_317", "Sap_m_318", "Sap_m_319", "Sap_m_320", "Sap_m_321",
    "Sap_m_322", "Sap_m_323", "Sap_m_324", "Sap_m_325", "Sap_m_326", "Sap_m_327",
    "Sap_m_328", "Sap_m_329", "Sap_m_330", "Sap_m_331", "Sap_m_332", "Sap_m_333",
    "Sap_m_334", "Sap_m_335", "Sap_m_336", "Sap_m_337", "Sap_m_338", "Sap_m_339",
    "Sap_m_340", "Sap_m_341", "Sap_m_342", "Sap_m_343", "Sap_m_344", "Sap_m_345",
    "Sap_m_346", "Sap_m_347", "Sap_m_348", "Sap_m_349", "Sap_m_350", "Sap_m_351",
    "Sap_m_352", "Sap_m_353", "Sap_m_354", "Sap_m_355", "Sap_m_356", "Sap_m_357",
    "Sap_m_358", "Sap_m_359", "Sap_m_360", "Sap_m_361", "Sap_m_362", "Sap_m_363",
    "Sap_m_364", "Sap_m_365", "Sap_m_366", "Sap_m_367", "Sap_m_368", "Sap_m_369",
    "Sap_m_370", "Sap_m_371", "Sap_m_372", "Sap_m_373", "Sap_m_374", "Sap_m_375",
    "Sap_m_376", "Sap_m_377", "Sap_m_378", "Sap_m_379", "Sap_m_380", "Sap_m_381",
    "Sap_m_382", "Sap_m_383", "Sap_m_384", "Sap_m_385", "Sap_m_386", "Sap_m_387",
    "Sap_m_388", "Sap_m_389", "Sap_m_390", "Sap_m_391", "Sap_m_392", "Sap_m_393",
    "Sap_m_394", "Sap_m_395", "Sap_m_396", "Sap_m_397", "Sap_m_398", "Sap_m_399",
    "Sap_m_400", "Sap_m_401", "Sap_m_402", "Sap_m_403", "Sap_m_404", "Sap_m_405",
    "Sap_m_406", "Sap_m_407", "Sap_m_408", "Sap_m_409", "Sap_m_410", "Sap_m_411",
    "Sap_m_412", "Sap_m_413", "Sap_m_414", "Sap_m_415", "Sap_m_416", "Sap_m_417",
    "Sap_m_418", "Sap_m_419", "Sap_m_420", "Sap_m_421", "Sap_m_422", "Sap_m_423",
    "Sap_m_424", "Sap_m_425", "Sap_m_426", "Sap_m_427", "Sap_m_428", "Sap_m_429",
    "Sap_m_430
```


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", "Venom_Allergy"] = 0
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_TP0",
    "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",
```

```

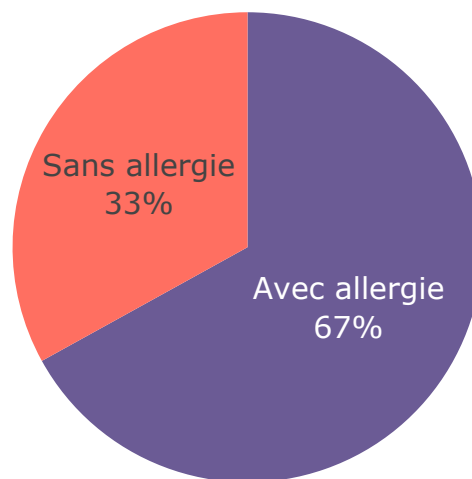
        "Avec allergie": "#6B5B95"
    }
)

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

```



Répartition des classes dans 'Allergy_Present'



```

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)

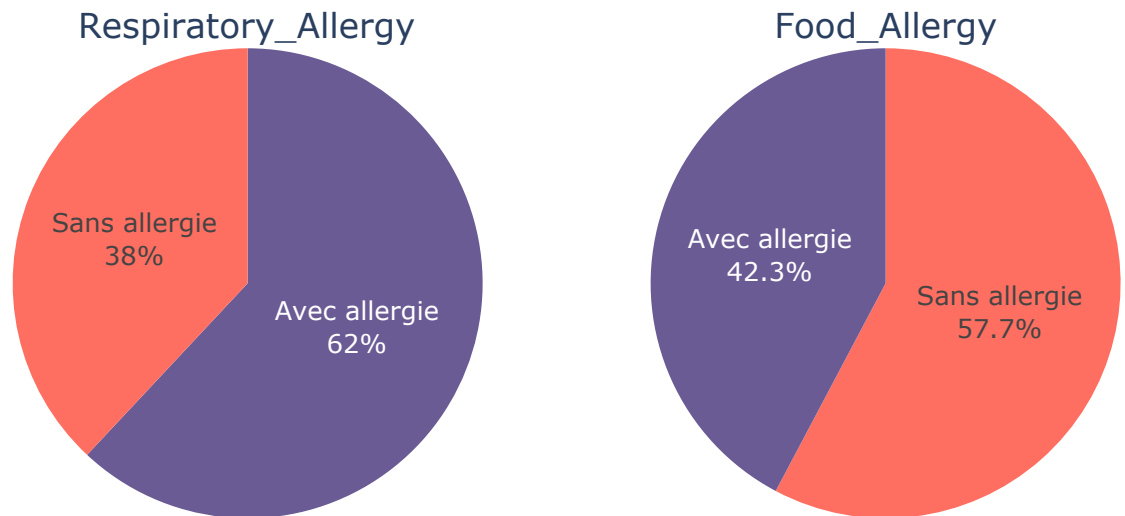
    fig.add_trace(
        go.Pie(
            labels=df_plot["Classe"],
            values=df_plot["Pourcentage"],
            name=col,
            marker=dict(colors=[colors_map[label] for label in df_plot["Classe"]
                               ],
                        textinfo="percent+label"
  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



Répartition des allergies par type



```
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)

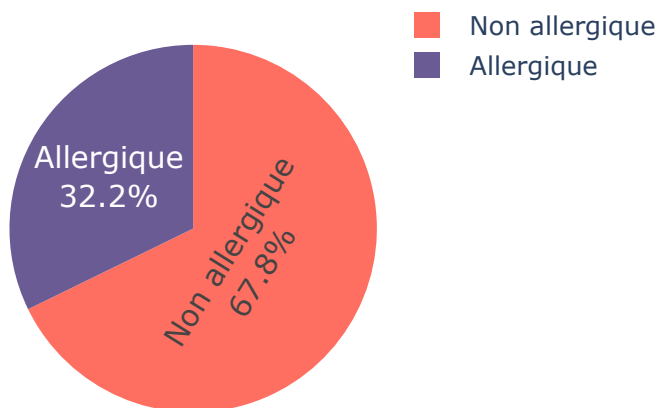
fig = px.pie(
    df_plot,
    names="Classe",
    values="Pourcentage",
    title=col,
    color="Classe",
    color_discrete_map=color_map
)

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

```

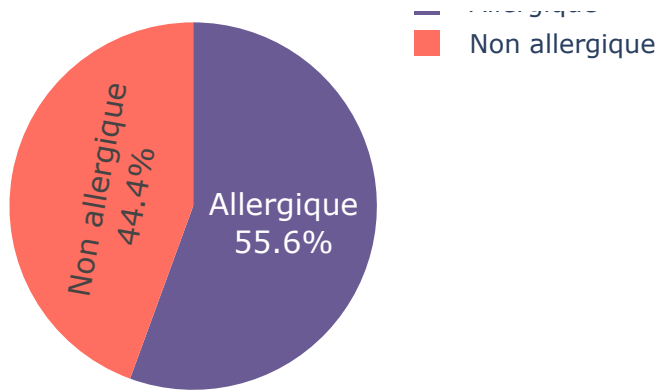


Type_of_Respiratory_Allergy_IGE_Pollen_Herb

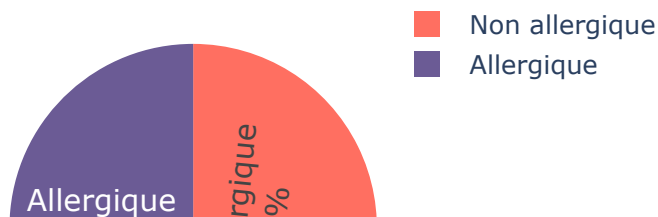


Type_of_Respiratory_Allergy_IGE_Pollen_Tree

■ 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)

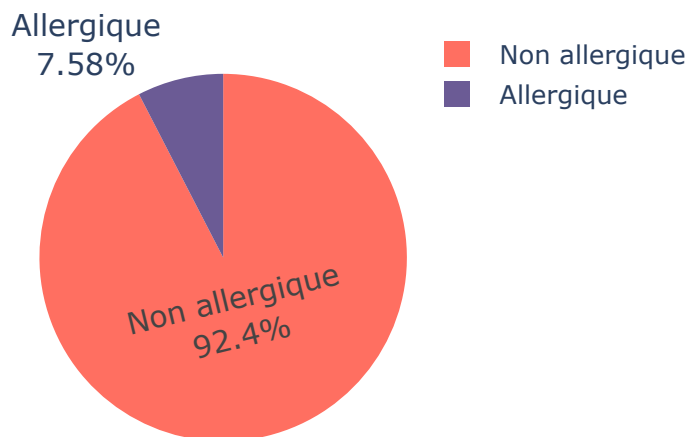
fig = px.pie(
    df_plot,
    names="Classe",
    values="Pourcentage",
    title=col,
    color="Classe",
    color_discrete_map=color_map
)

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

```

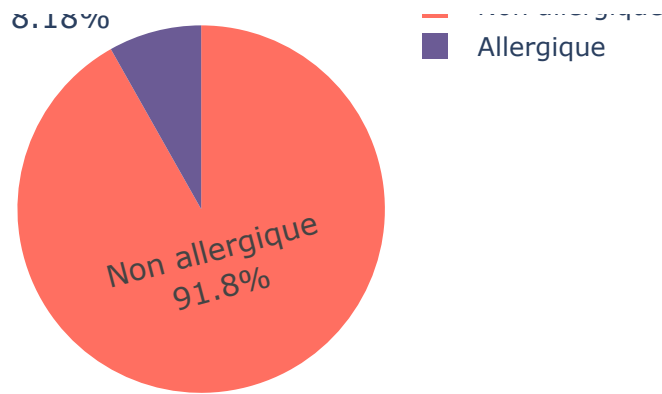


Type_of_Food_Allergy_Aromatics



Type_of_Food_Allergy_Cereals_&_Seeds



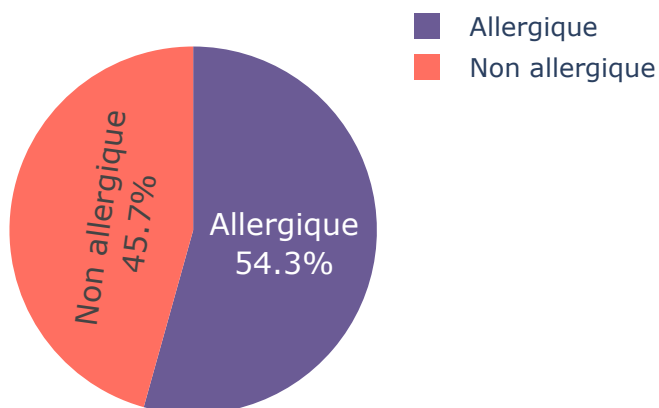


Type_of_Food_Allergy_Egg



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_TP0",
    "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

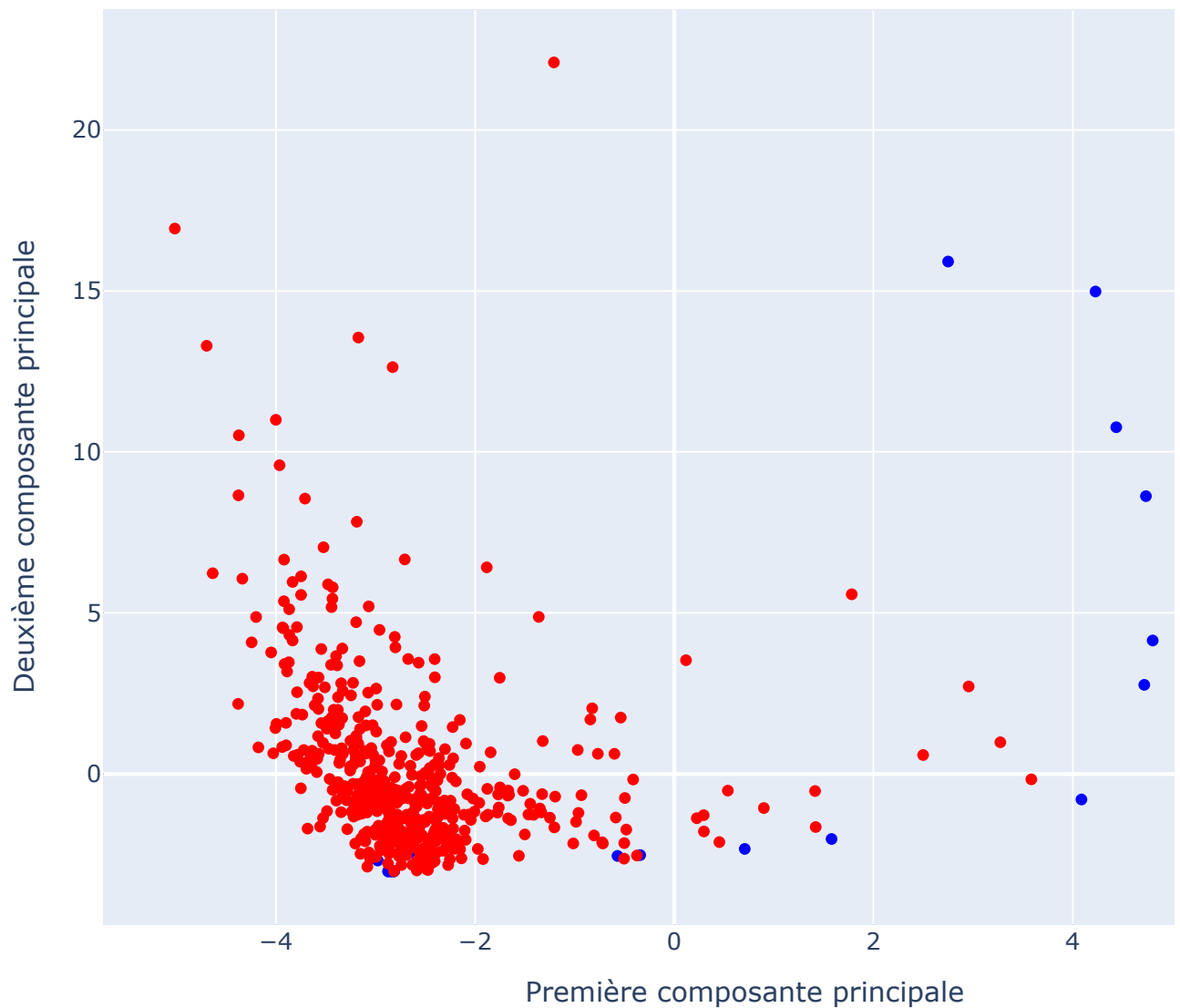
# Tracé du graphique
fig = px.scatter(
    df_pca,
    x="PC1",
    y="PC2",
    color=df_pca["Allergy_Present"].astype(str), # couleur par classe 0 ou 1
    title="Projection PCA des patients",
    labels={
        "PC1": "Première composante principale",
        "PC2": "Deuxième composante principale",
        "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()
```



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 = 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)

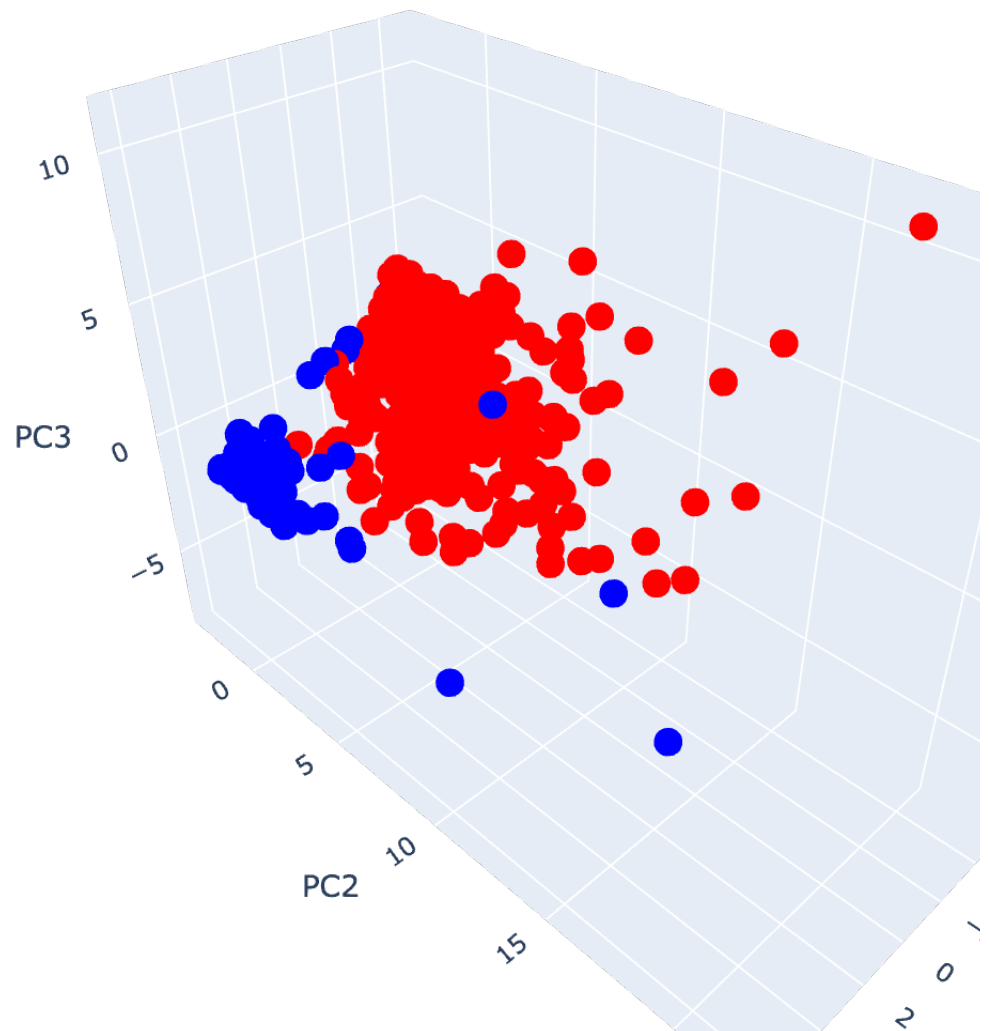
# 4. Reconstruction du DataFrame
df_pca = pd.DataFrame(X_pca, columns=["PC1", "PC2", "PC3"])
df_pca["Allergy_Present"] = y.values

# 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

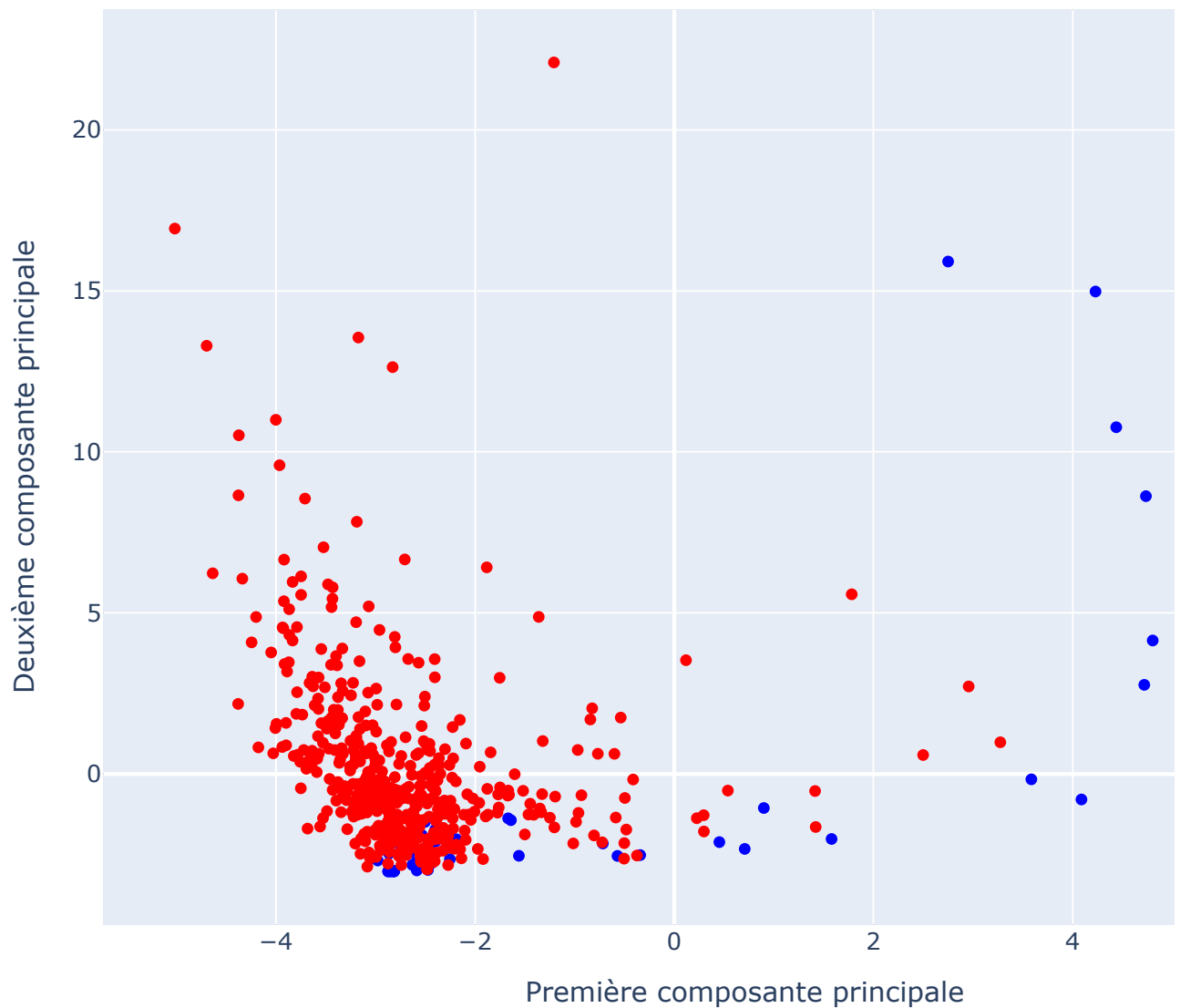
# Tracé du graphique
fig = px.scatter(
    df_pca,
    x="PC1",
    y="PC2",
    color=df_pca["Respiratory_Allergy"].astype(str), # couleur par classe 0 ou 1
    title="Projection PCA des patients",
    labels={
        "PC1": "Première composante principale",
        "PC2": "Deuxième composante principale",
        "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
```

```
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)

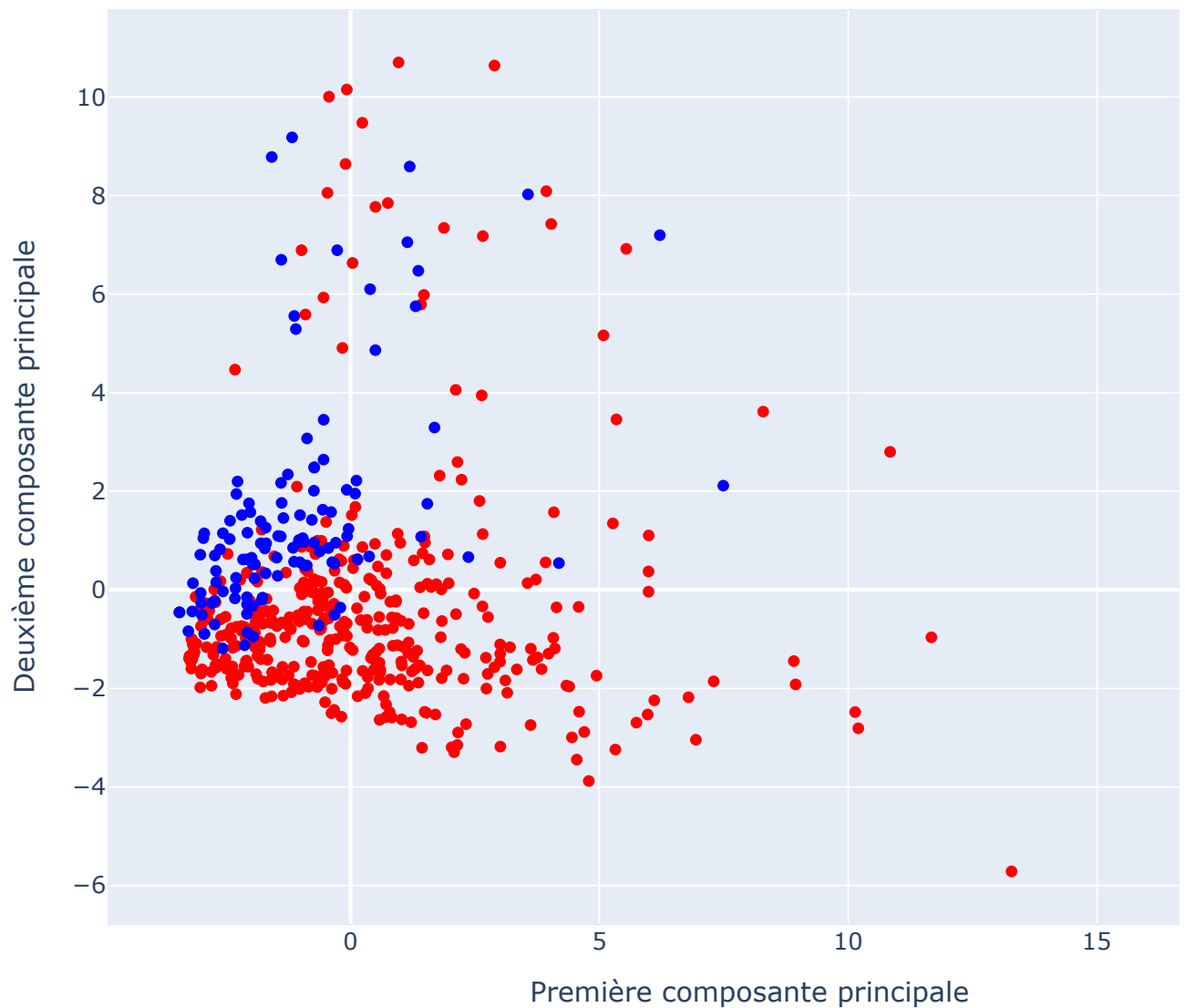
df_pca = pd.DataFrame(X_pca, columns=["PC1", "PC2"])
df_pca["Severe_Allergy"] = y.values # Ajouter la cible

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()
```




Projection PCA des patients



```
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

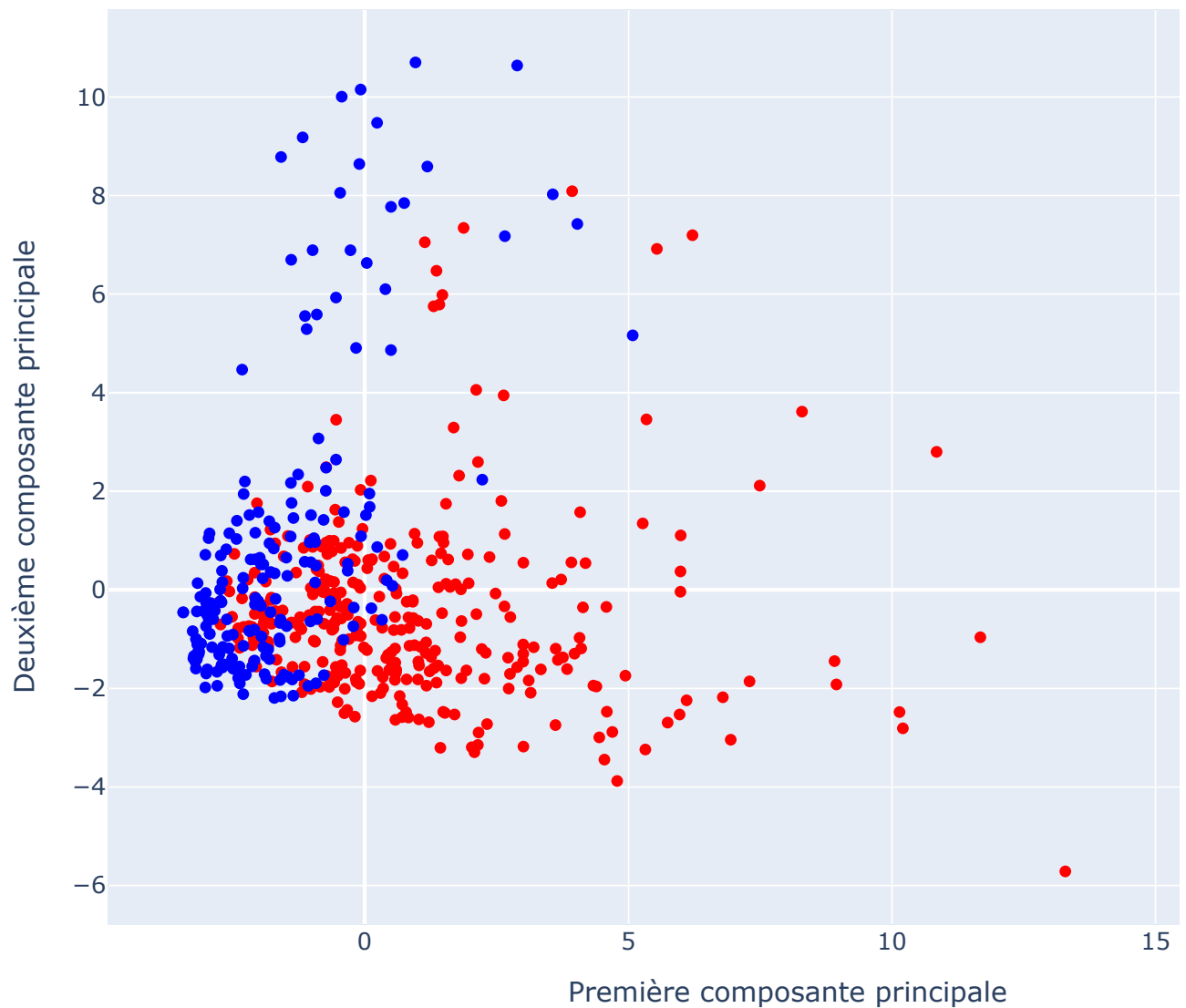
# Tracé du graphique
fig = px.scatter(
    df_pca,
    x="PC1",
    y="PC2",
    color=df_pca["Food_Allergy"].astype(str), # couleur par classe 0 ou 1
    title="Projection PCA des patients",
    labels={
        "PC1": "Première composante principale",
        "PC2": "Deuxième composante principale",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

fig.update_layout(
    legend_title="Food Allergie"
)

fig.show()
```



Projection PCA des patients



```
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

