Imports

```
#Sickit learn met régulièrement à jour des versions et
#indique des futurs warnings.
#ces deux lignes permettent de ne pas les afficher.
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
# librairies générales
import pandas as pd
import re
from tabulate import tabulate
import time
import numpy as np
import pickle
import string
import base64
import sys
# librairie affichage
import matplotlib.pyplot as plt
import seaborn as sns
# librairies scikit learn
import sklearn
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.base import TransformerMixin
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
# librairies des classifiers utilisés
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
import warnings
warnings.filterwarnings("ignore")
```

```
# pour monter son drive Google Drive local
from google.colab import drive
drive.mount('/content/gdrive')
my_local_drive='/content/gdrive/My Drive/Colab Notebooks/TER'
# Ajout du path pour les librairies, fonctions et données
sys.path.append(my_local_drive)
# Se positionner sur le répertoire associé
%cd $my_local_drive
%pwd
```

```
Mounted at /content/gdrive
/content/gdrive/My Drive/Colab Notebooks/TER
'/content/gdrive/My Drive/Colab Notebooks/TER'
```

Classification

```
ALEX = pd.read_excel("ALEX.xlsx")
ALEX.head()
```

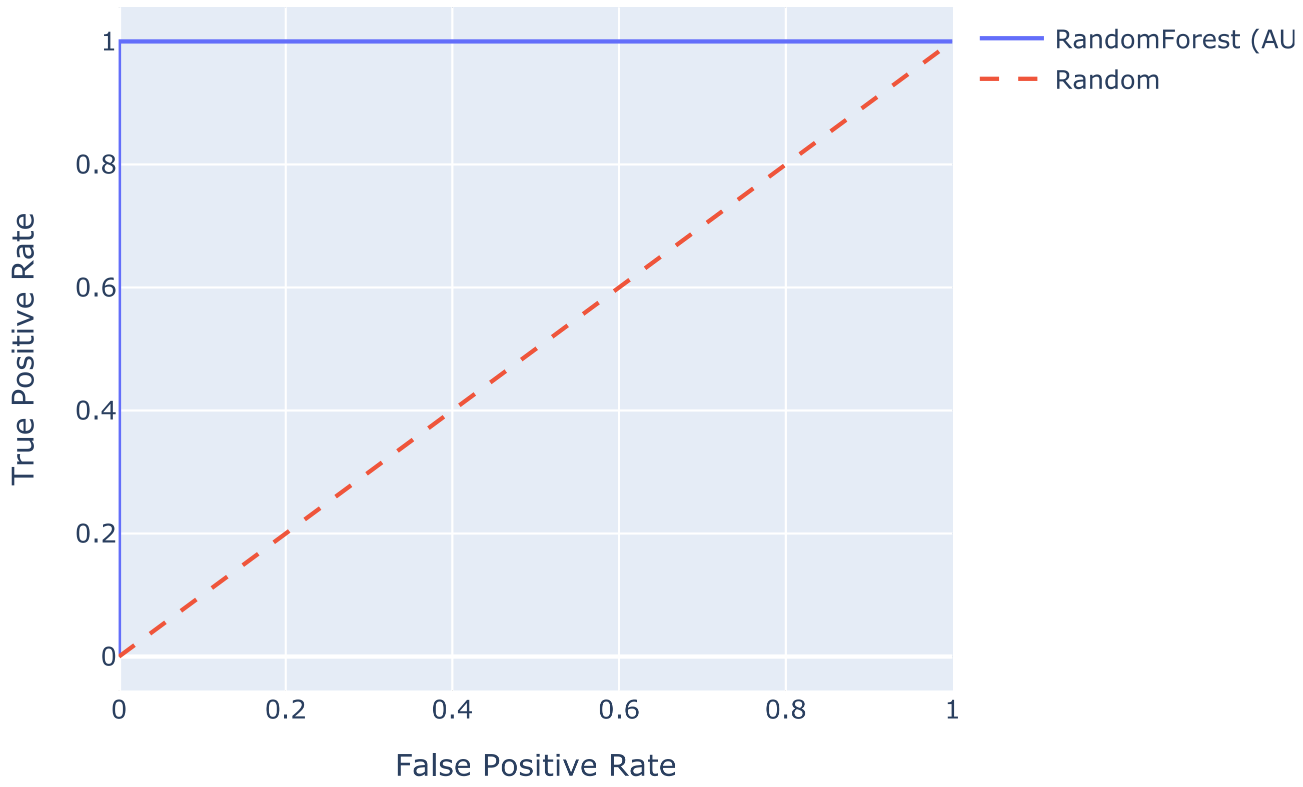
	Unnamed: 0	Chip_Code	Chip_Type	Age	Gender	French_Residence_Department
0	PMP0237	02AGT832	ALEX	28	F	999
1	PMP0238	02AGT834	ALEX	20	M	999
2	PMP0239	02AGT835	ALEX	22	F	999
3	PMP0240	02AGT486	ALEX	10	M	999
4	PMP0241	02AGT488	ALEX	2	M	999

5 rows × 138 columns

```
target_1 = [
    "Allergy_Present",
    "Respiratory_Allergy",
    "Food_Allergy",
    "Venom_Allergy",
    "Severe_Allergy",
    "Type_of_Food_Allergy_Other",
    "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",
    "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",
    "Type_of_Venom_Allergy_ATCD_Venom",
    "Type_of_Venom_Allergy_IGE_Venom",
]

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

extra = ['History_of_food_anaphylaxis','First_degree_family_history_of_atopy',
         'History_of_hymenoptera_venom_anaphylaxis','Mammalian_meat']
extra_1 = ["Conjunctivitis", "Oral_Syndrom", "Cardiovascular_symptoms", "Respir

Gina = ["GINA_(asthma)_0", "GINA_(asthma)_1", "GINA_(asthma)_2", "GINA_(asthma)
inconnu = ["Treatment_of_athsma_9", "Treatment_of_rhinitis_9", "General_cofacto
           "Age_of_onsets_9", "ARIA_(rhinitis)_9", "GINA_(asthma)_9", "Treatmen
Aria = ["ARIA_(rhinitis)_9", "ARIA_(rhinitis)_0", "ARIA_(rhinitis)_1", "ARIA_(r


import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedKFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.metrics import (
    f1_score, accuracy_score, recall_score,
    precision_score, confusion_matrix, roc_auc_score, roc_curve
)
from imblearn.over_sampling import SMOTE
import plotly.graph_objects as go

targets = ["Allergy_Present", "Respiratory_Allergy", "Food_Allergy", "Venom_All

models = {
    "RandomForest": RandomForestClassifier(random_state=42),
    "XGBoost": XGBClassifier(random_state=42, eval_metric="logloss", use_label_
    "LogisticRegression": LogisticRegression(max_iter=1000, random_state=42),
    "SVM": SVC(probability=True, random_state=42)
}

X = ALEX.copy()
X.drop(target_1, axis=1, inplace=True)
X.drop(extra_columns, axis=1, inplace=True)
X.drop(extra, axis=1, inplace=True)
X.drop(inconnu, axis=1, inplace=True)
X = X.iloc[:, 1:]

results = []
kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

for target in targets:
    y = ALEX[target]

    for model_name, base_model in models.items():
        f1_class0_scores, f1_class1_scores = [], []
        precision_scores, acc_scores, recall_scores, auc_scores = [], [], [], [

        for train_idx, test_idx in kfold.split(X, y):
            X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
            y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]

            smote = SMOTE(random_state=42)
            X_train_res, y_train_res = smote.fit_resample(X_train, y_train)

            base_model.fit(X_train_res, y_train_res)
            y_pred = base_model.predict(X_test)

            acc_scores.append(accuracy_score(y_test, y_pred))
            recall_scores.append(recall_score(y_test, y_pred, zero_division=0))
            precision_scores.append(precision_score(y_test, y_pred, average='we
            f1_class0_scores.append(f1_score(y_test, y_pred, pos_label=0, zero_
            f1_class1_scores.append(f1_score(y_test, y_pred, pos_label=1, zero_

            if hasattr(base_model, "predict_proba"):
                y_proba = base_model.predict_proba(X_test)[:, 1]
                auc_scores.append(roc_auc_score(y_test, y_proba))

        base_model.fit(X, y)
        y_pred_full = base_model.predict(X)
        y_proba_full = base_model.predict_proba(X)[:, 1] if hasattr(base_model,
        matrix = confusion_matrix(y, y_pred_full)

        print(f"\n🔍 Target: {target} | Model: {model_name}")
        print(f"📈 Accuracy: {np.mean(acc_scores):.4f}")
        print(f"🎯 F1 (0): {np.mean(f1_class0_scores):.4f} | F1 (1): {np.mean(
        print(f"📊 Precision: {np.mean(precision_scores):.4f} | AUC: {np.mean(
        print("📊 Confusion Matrix:\n", matrix)

        if y_proba_full is not None:
            fpr, tpr, _ = roc_curve(y, y_proba_full)
            fig = go.Figure()
            fig.add_trace(go.Scatter(x=fpr, y=tpr, mode='lines', name=f"{model_
            fig.add_trace(go.Scatter(x=[0, 1], y=[0, 1], mode='lines', name='Ra
            fig.update_layout(
                title=f"ROC Curve - {target} - {model_name}",
                xaxis_title="False Positive Rate",
                yaxis_title="True Positive Rate",
                width=700, height=500
            )
            fig.show()

        results.append({
            "Target": target,
            "Model": model_name,
            "F1_Class_0": np.mean(f1_class0_scores),
            "F1_Class_1": np.mean(f1_class1_scores),
            "Precision": np.mean(precision_scores),
            "Accuracy": np.mean(acc_scores),
            "Recall": np.mean(recall_scores),
            "AUC_ROC": np.mean(auc_scores) if auc_scores else np.nan
        })

pd.DataFrame(results).to_csv("results_ALEX_Allergie.csv", index=False)
```

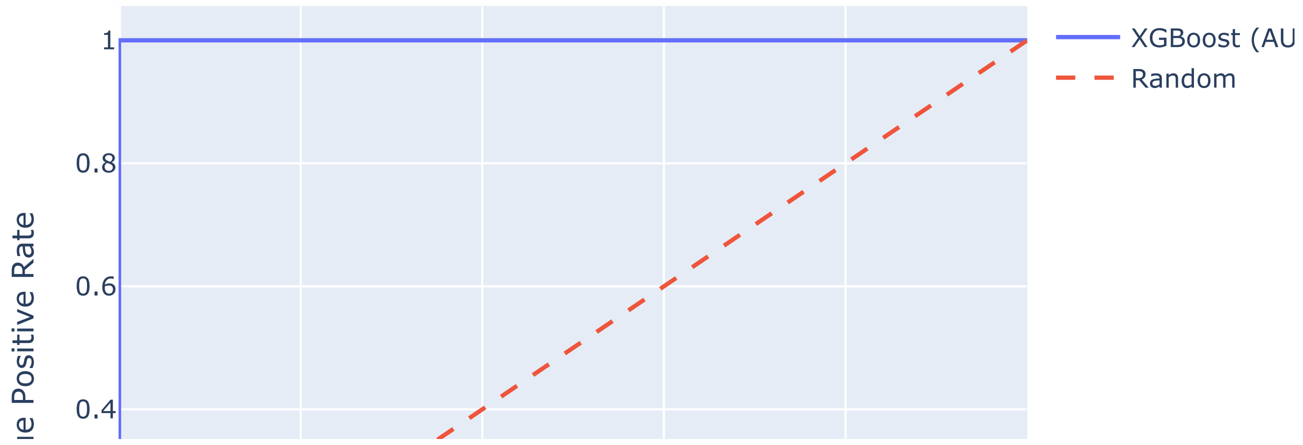
```
🔍 Target: Allergy_Present | Model: RandomForest
📈 Accuracy: 0.9710
🎯 F1 (0): 0.9701 | F1 (1): 0.9718
📊 Precision: 0.9717 | AUC: 0.9954517180656864
📊 Confusion Matrix:
 [[554   0]
 [  0 585]]
```

ROC Curve - Allergy_Present - RandomForest



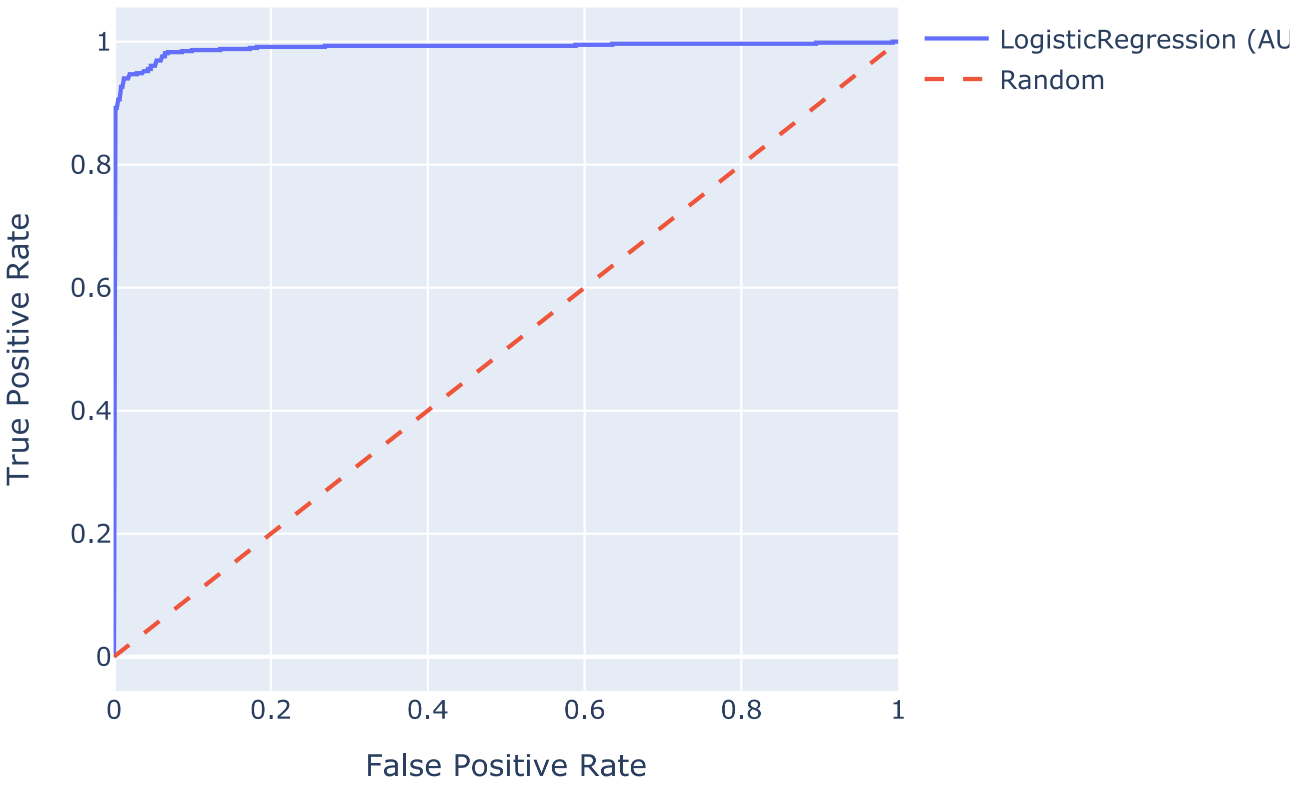
```
🔍 Target: Allergy_Present | Model: XGBoost
📈 Accuracy: 0.9605
🎯 F1 (0): 0.9595 | F1 (1): 0.9614
📊 Precision: 0.9611 | AUC: 0.990096216612143
📊 Confusion Matrix:
 [[554   0]
 [  0 585]]
```

ROC Curve - Allergy_Present - XGBoost

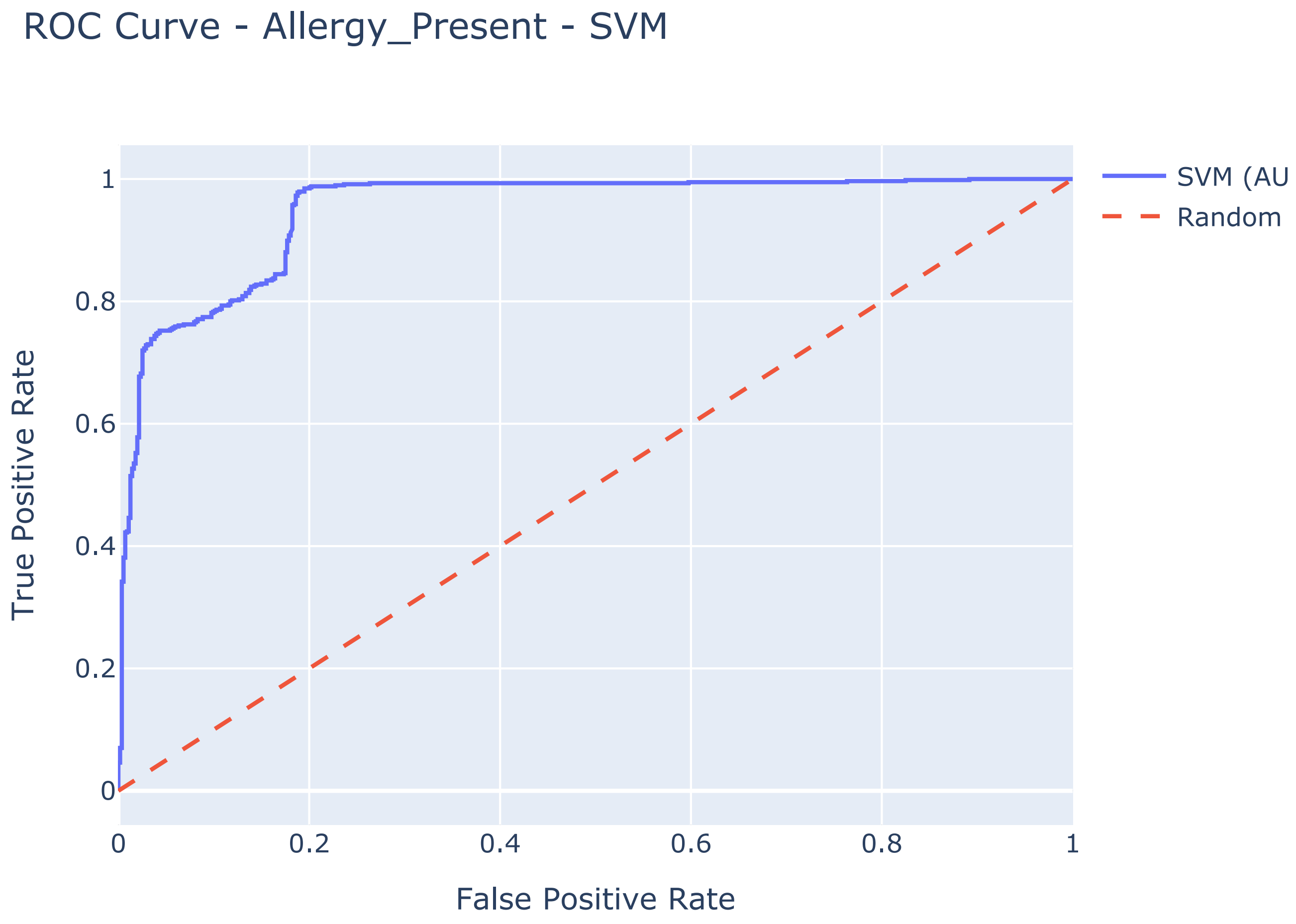


```
🔍 Target: Allergy_Present | Model: LogisticRegression
📈 Accuracy: 0.9464
🎯 F1 (0): 0.9461 | F1 (1): 0.9467
📊 Precision: 0.9485 | AUC: 0.981369367044411
📊 Confusion Matrix:
 [[544  10]
 [ 33 552]]
```

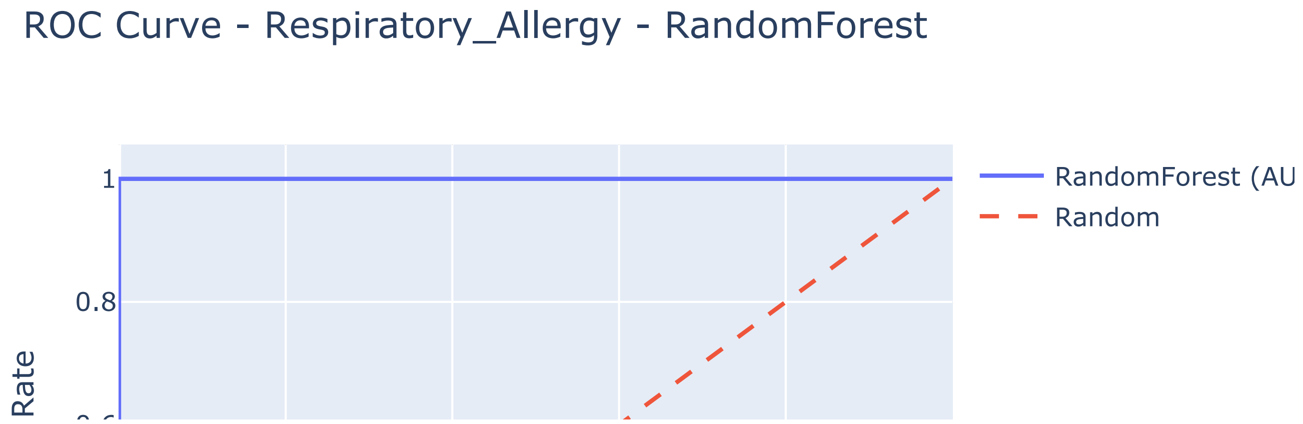
ROC Curve - Allergy_Present - LogisticRegression



```
🔍 Target: Allergy_Present | Model: SVM
📈 Accuracy: 0.8709
🎯 F1 (0): 0.8595 | F1 (1): 0.8804
📊 Precision: 0.8753 | AUC: 0.9319373021776587
📊 Confusion Matrix:
 [[453 101]
 [ 39 546]]
```

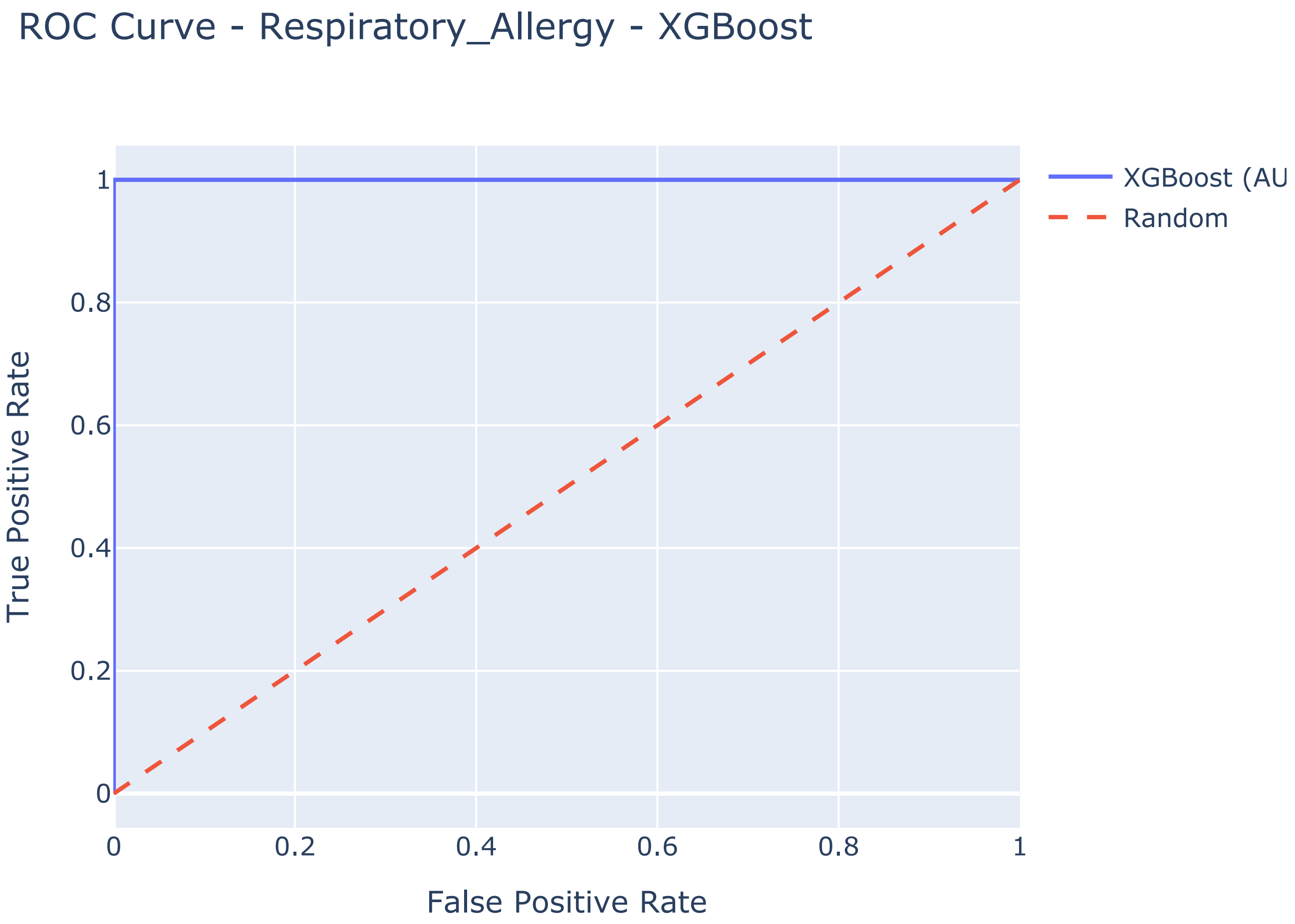


```
🔍 Target: Respiratory_Allergy | Model: RandomForest
📈 Accuracy: 0.9517
🎯 F1 (0): 0.9568 | F1 (1): 0.9451
📊 Precision: 0.9530 | AUC: 0.9890913952119309
📊 Confusion Matrix:
 [[644   0]
 [  0 495]]
```

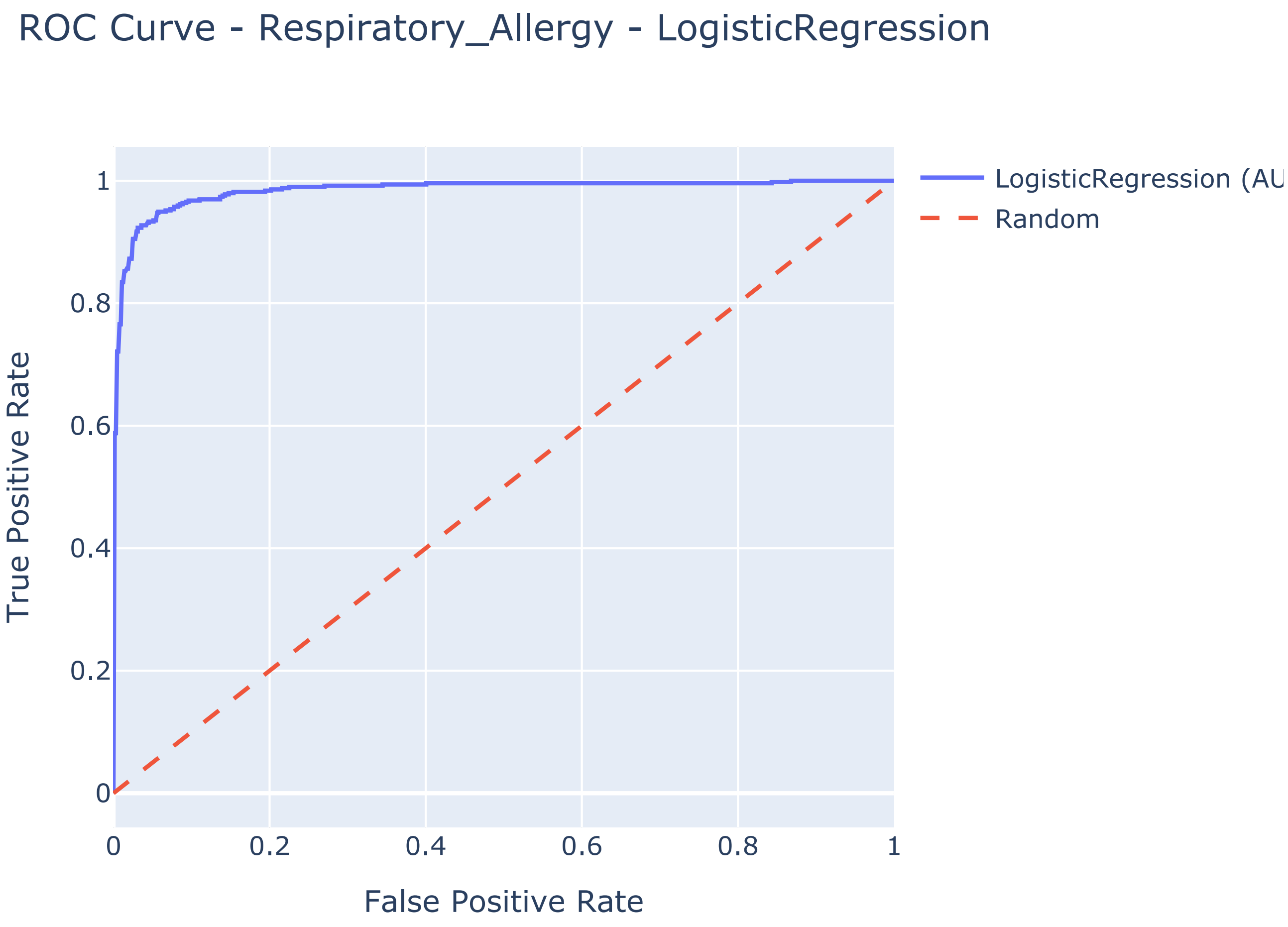


True Positive
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
False Positive Rate

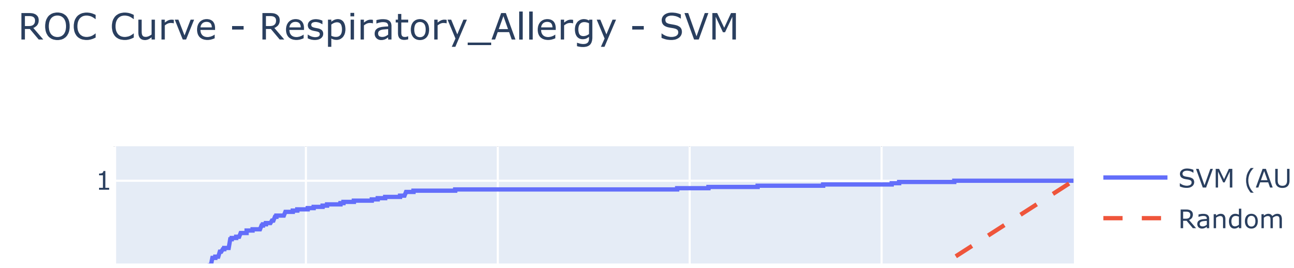
```
🔍 Target: Respiratory_Allergy | Model: XGBoost
📈 Accuracy: 0.9543
🎯 F1 (0): 0.9599 | F1 (1): 0.9470
📊 Precision: 0.9551 | AUC: 0.986819122841444
📊 Confusion Matrix:
 [[644   0]
 [  0 495]]
```

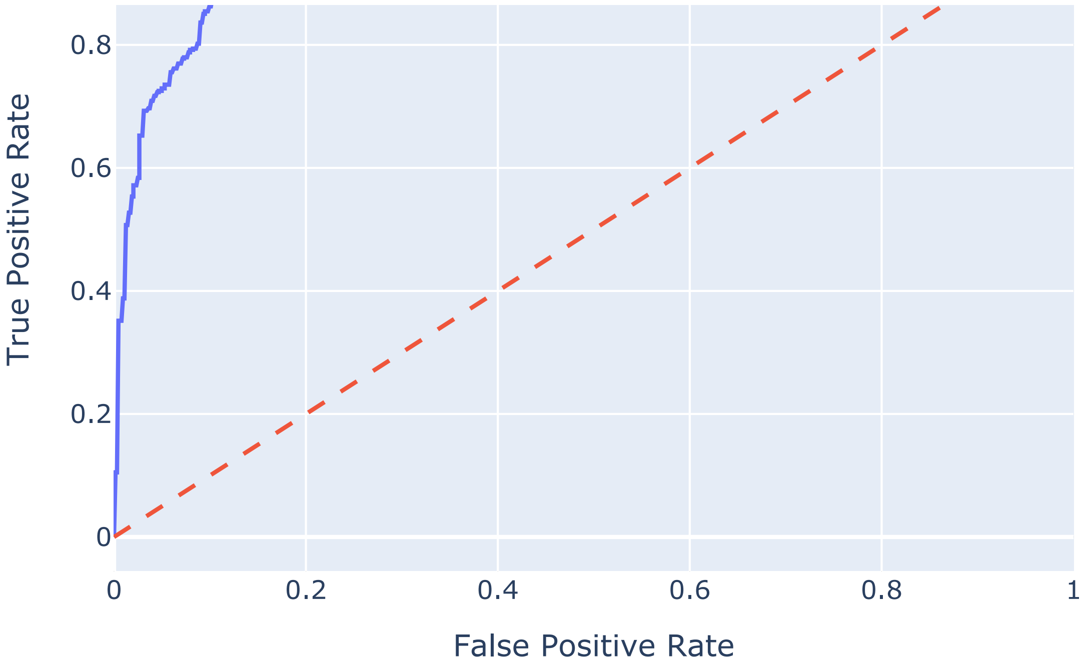


```
🔍 Target: Respiratory_Allergy | Model: LogisticRegression
📈 Accuracy: 0.9271
🎯 F1 (0): 0.9356 | F1 (1): 0.9158
📊 Precision: 0.9287 | AUC: 0.970541159733124
📊 Confusion Matrix:
 [[624  20]
 [ 38 457]]
```

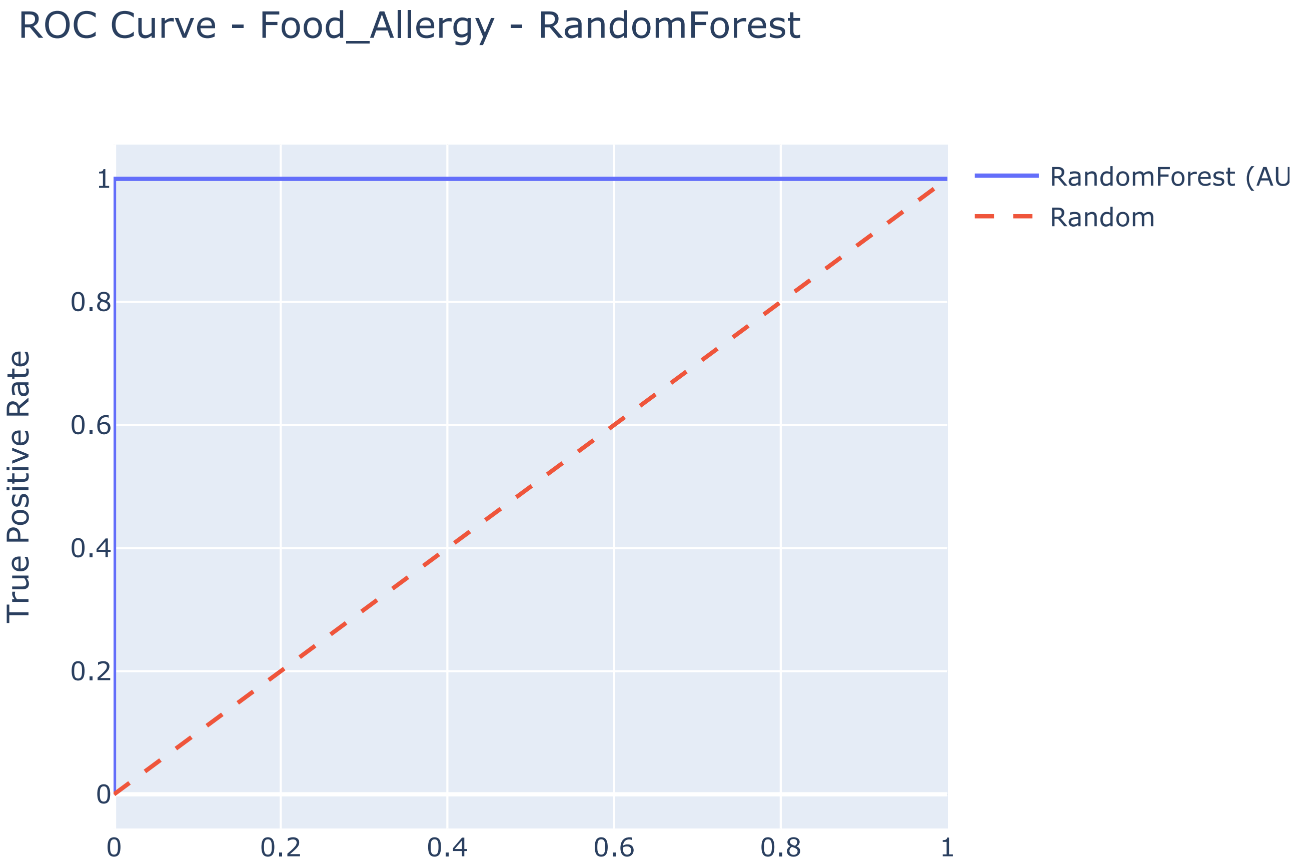


```
🔍 Target: Respiratory_Allergy | Model: SVM
📈 Accuracy: 0.8095
🎯 F1 (0): 0.8122 | F1 (1): 0.8062
📊 Precision: 0.8323 | AUC: 0.9098046507064363
📊 Confusion Matrix:
 [[607  37]
 [125 370]]
```





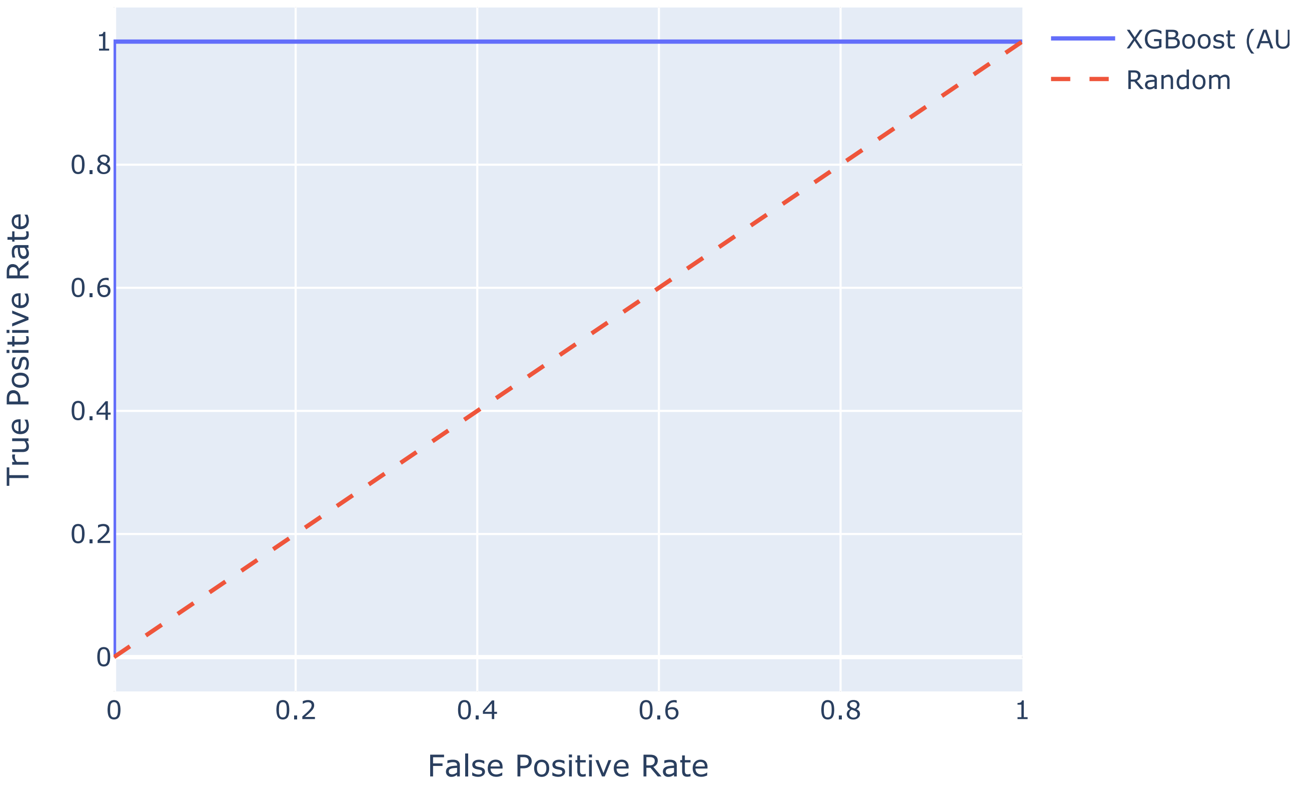
```
🔍 Target: Food_Allergy | Model: RandomForest
📈 Accuracy: 0.9035
🎯 F1 (0): 0.9251 | F1 (1): 0.8641
📊 Precision: 0.9087 | AUC: 0.9690984444917963
📊 Confusion Matrix:
 [[753   0]
 [  0 386]]
```



False Positive Rate

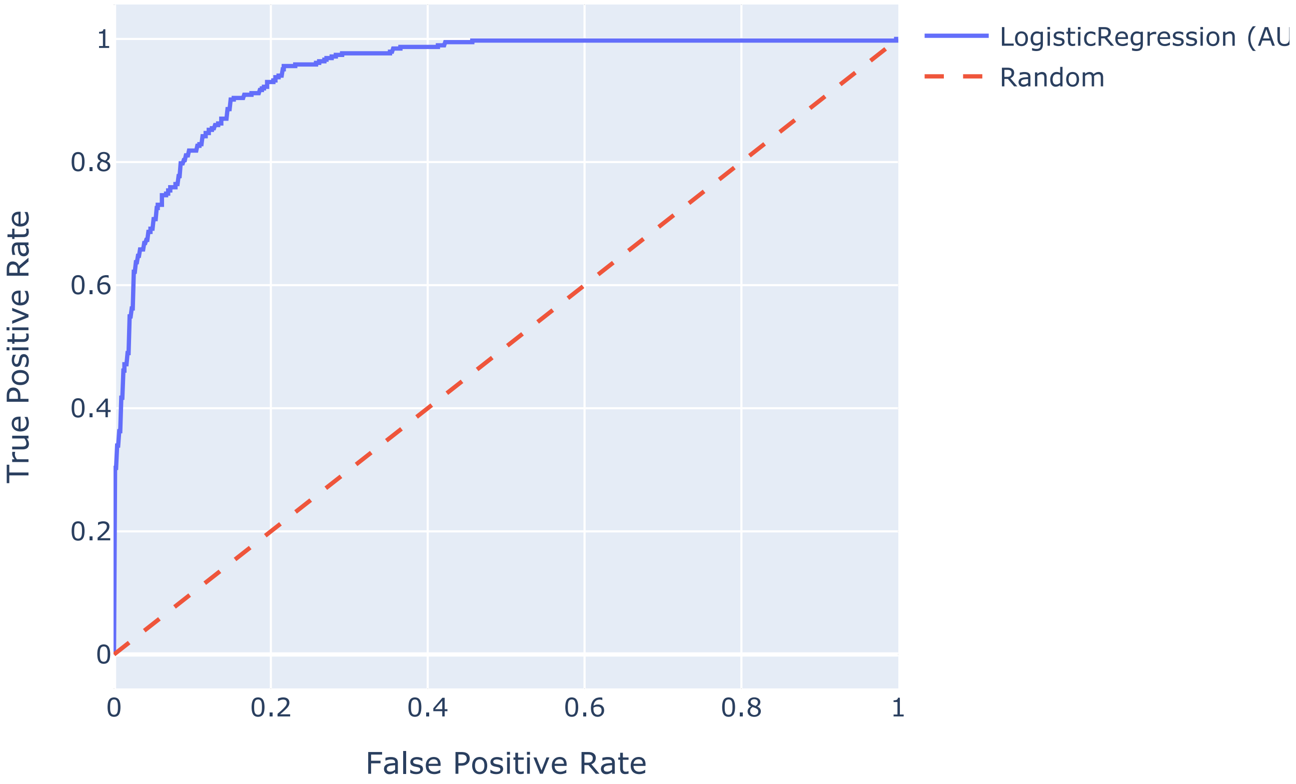
```
🔍 Target: Food_Allergy | Model: XGBoost
📈 Accuracy: 0.9149
🎯 F1 (0): 0.9356 | F1 (1): 0.8739
📊 Precision: 0.9162 | AUC: 0.9713629282382745
📊 Confusion Matrix:
 [[753   0]
 [  0 386]]
```

ROC Curve - Food_Allergy - XGBoost

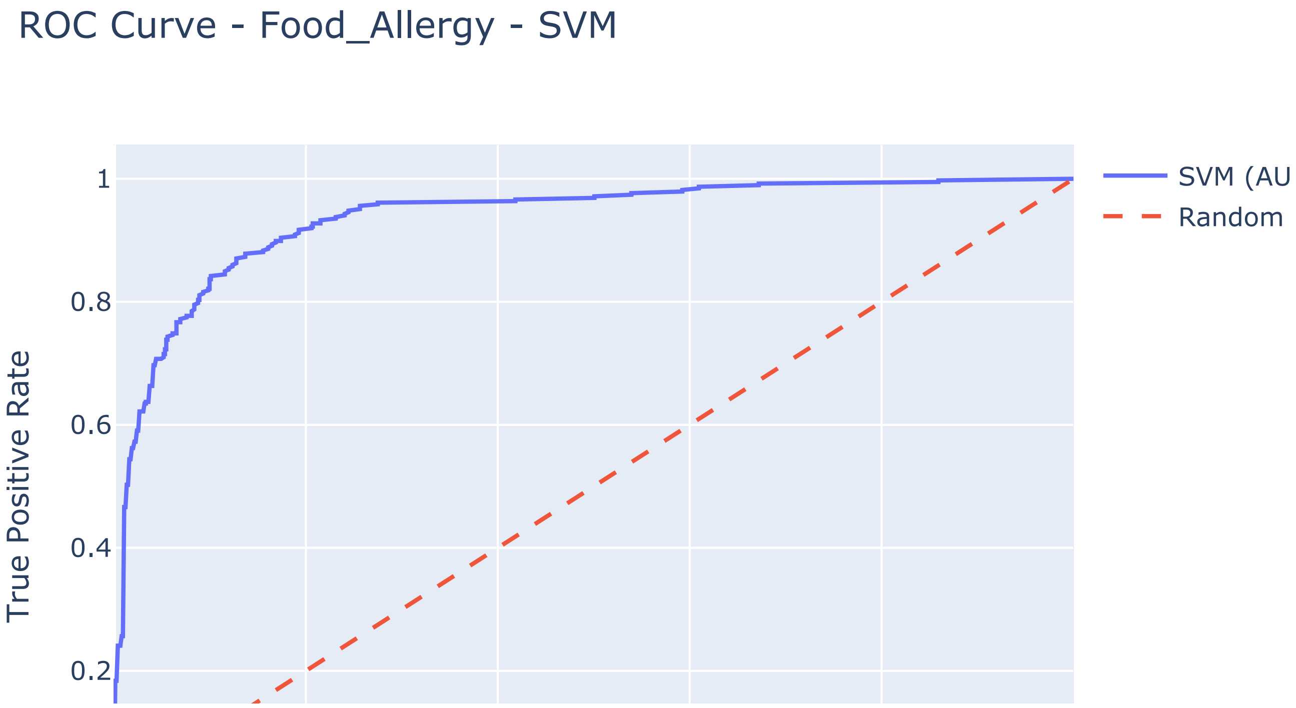


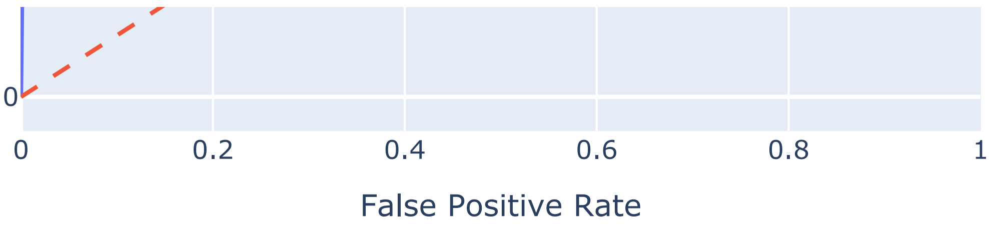
```
🔍 Target: Food_Allergy | Model: LogisticRegression
📈 Accuracy: 0.8437
🎯 F1 (0): 0.8796 | F1 (1): 0.7762
📊 Precision: 0.8487 | AUC: 0.9080047351847907
📊 Confusion Matrix:
 [[687  66]
 [ 77 309]]
```

ROC Curve - Food_Allergy - LogisticRegression

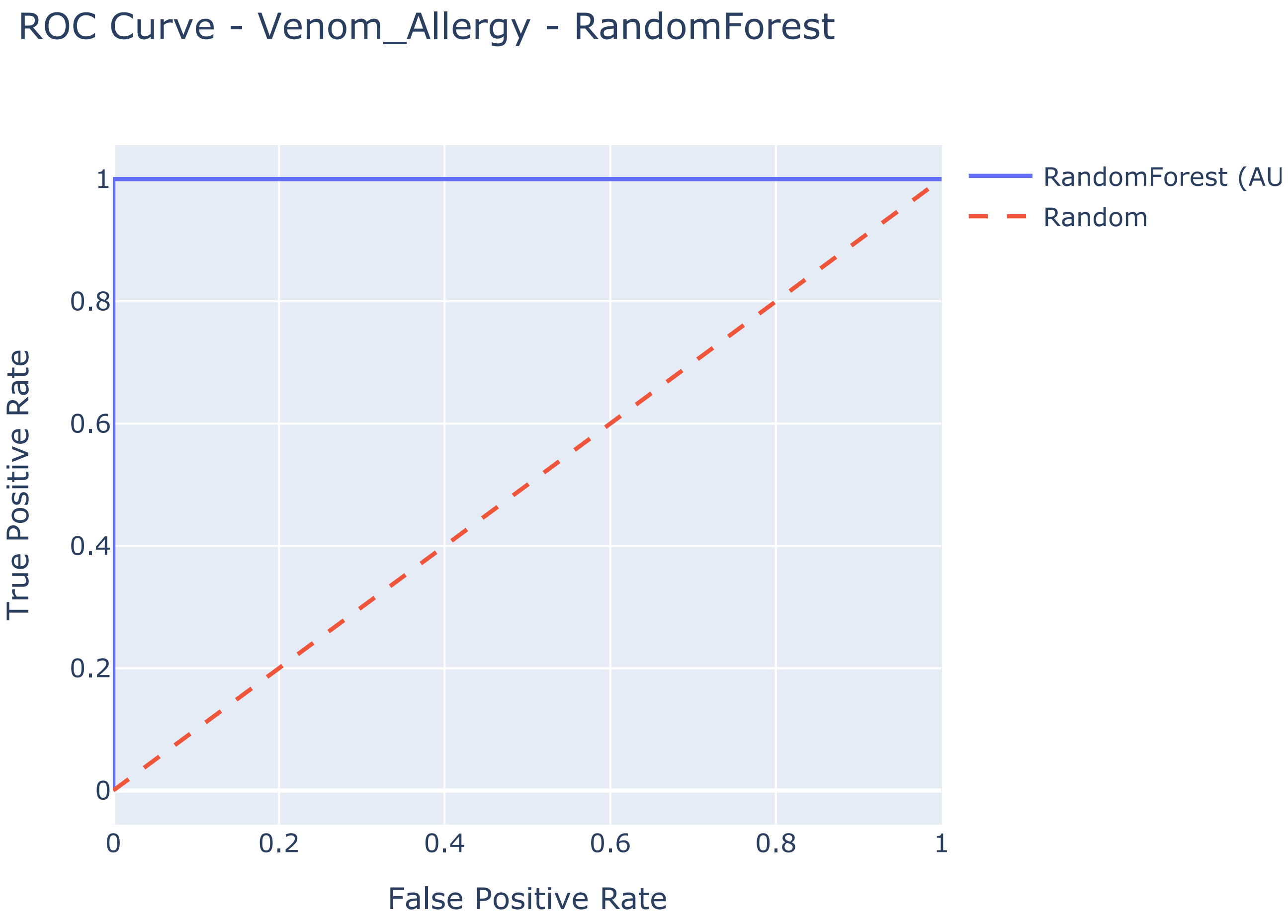


```
🔍 Target: Food_Allergy | Model: SVM
📈 Accuracy: 0.8420
🎯 F1 (0): 0.8804 | F1 (1): 0.7659
📊 Precision: 0.8436 | AUC: 0.9047985534010465
📊 Confusion Matrix:
 [[726  27]
 [139 247]]
```

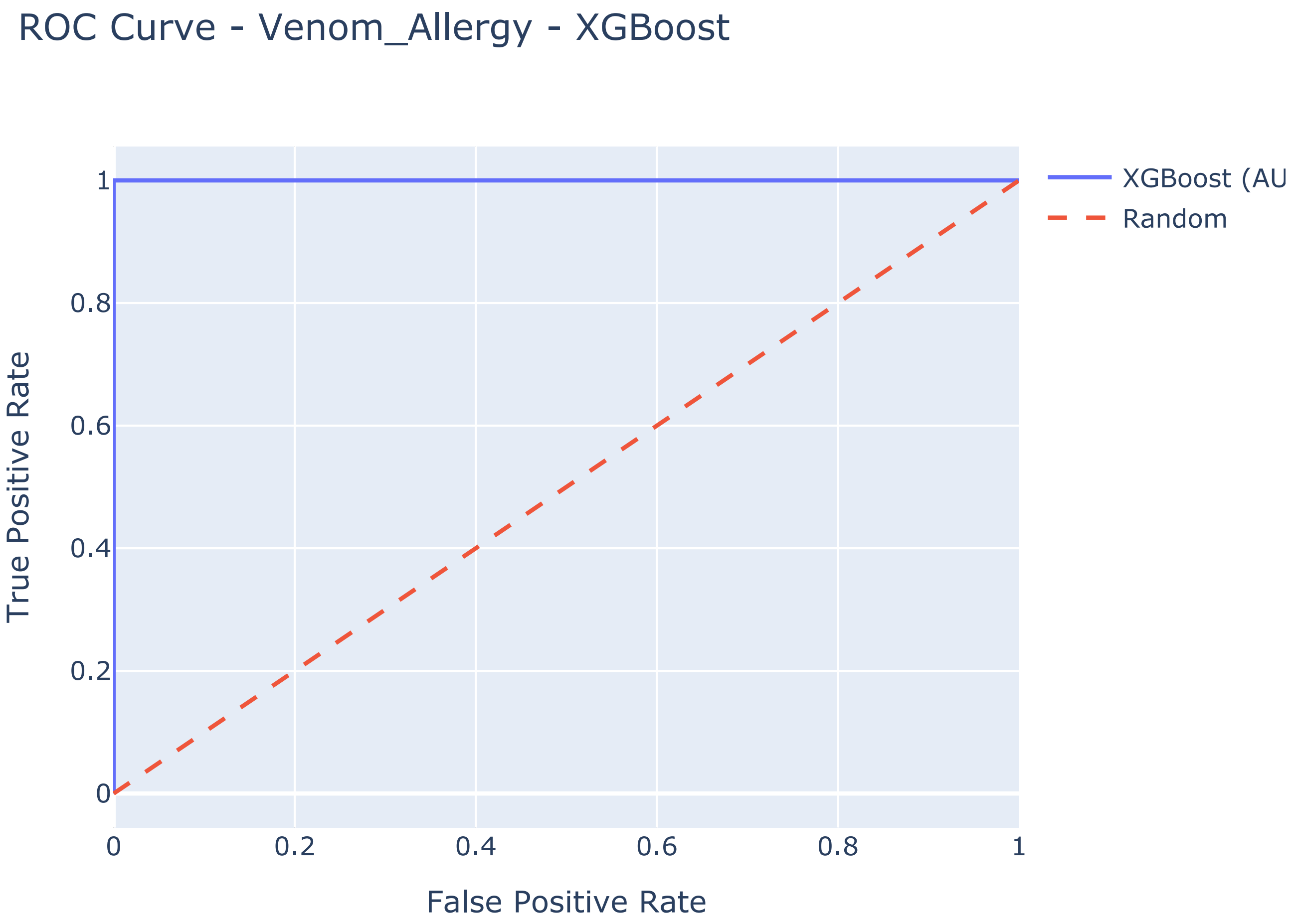




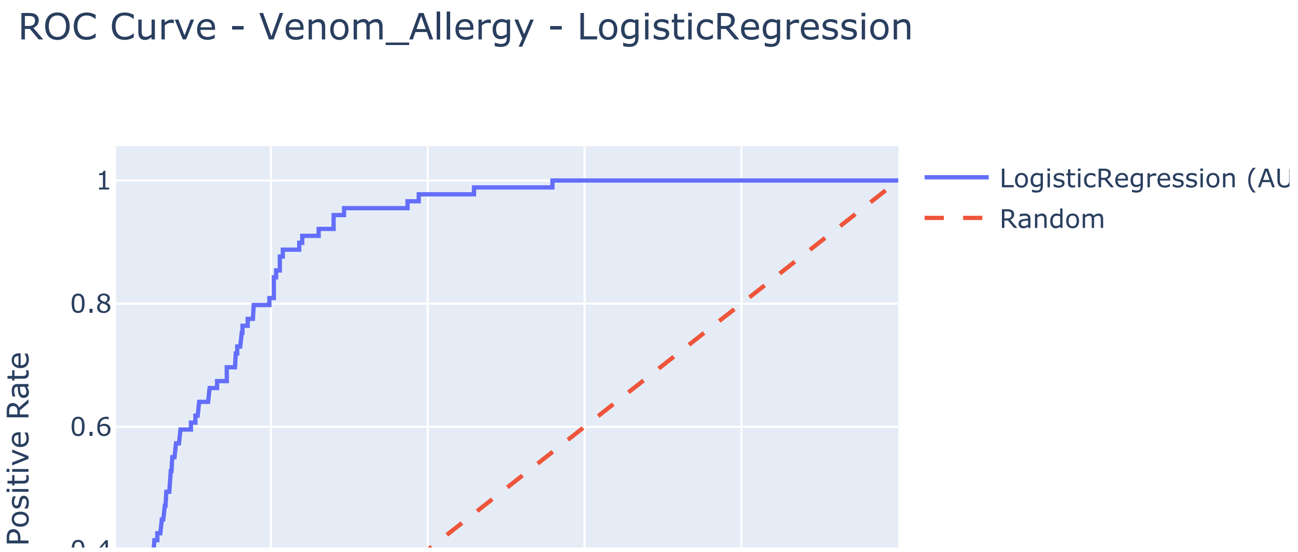
```
🔍 Target: Venom_Allergy | Model: RandomForest
📈 Accuracy: 0.9105
🎯 F1 (0): 0.9518 | F1 (1): 0.3513
📊 Precision: 0.9023 | AUC: 0.8839550264550265
📊 Confusion Matrix:
 [[1050    0]
 [   0   89]]
```



```
🔍 Target: Venom_Allergy | Model: XGBoost
📈 Accuracy: 0.9061
🎯 F1 (0): 0.9491 | F1 (1): 0.3829
📊 Precision: 0.9049 | AUC: 0.8575264550264551
📊 Confusion Matrix:
 [[1050    0]
 [   0   89]]
```

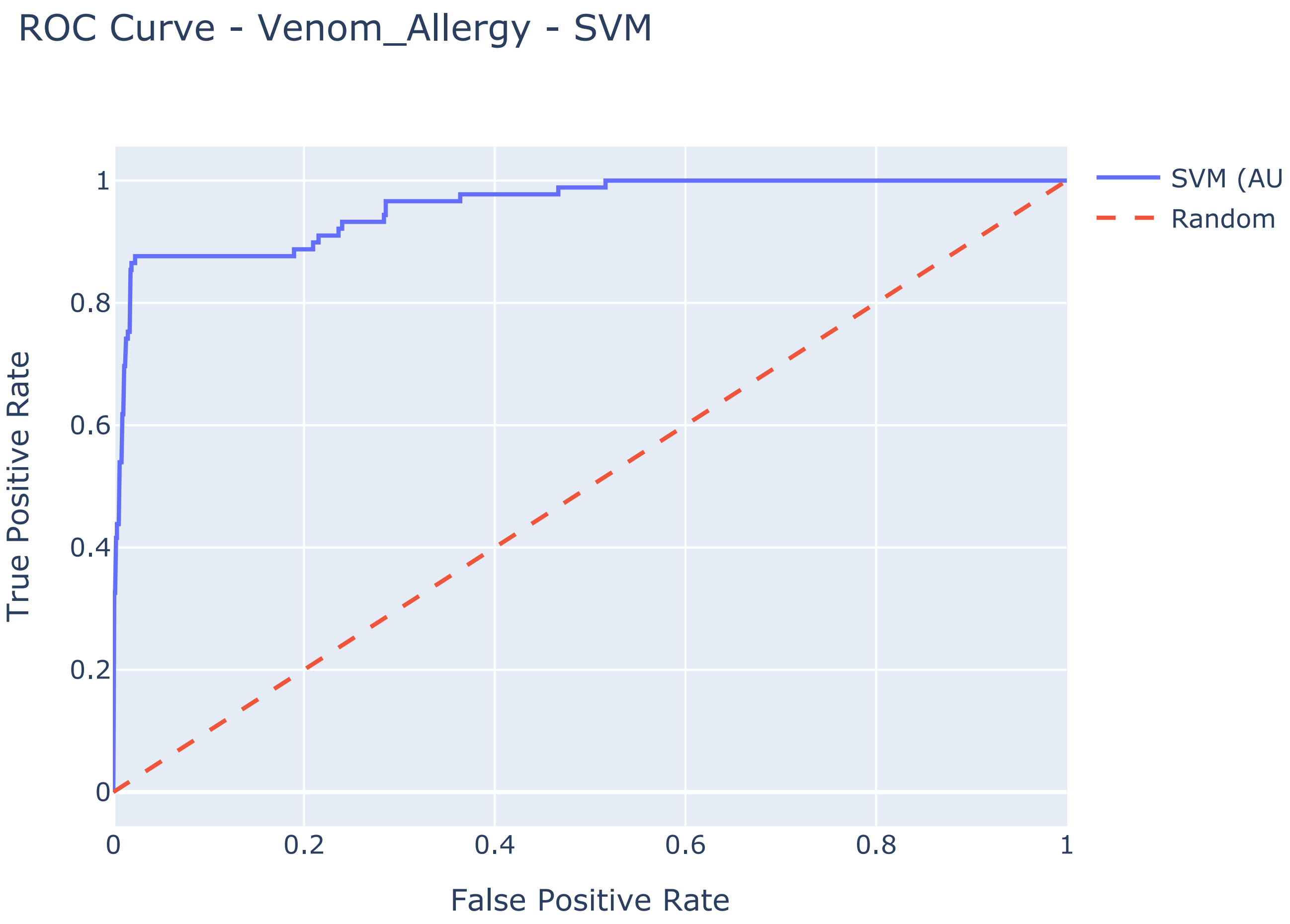


```
🔍 Target: Venom_Allergy | Model: LogisticRegression
📈 Accuracy: 0.8516
🎯 F1 (0): 0.9174 | F1 (1): 0.2575
📊 Precision: 0.8837 | AUC: 0.6994444444444443
📊 Confusion Matrix:
 [[1043    7]
 [  77   12]]
```



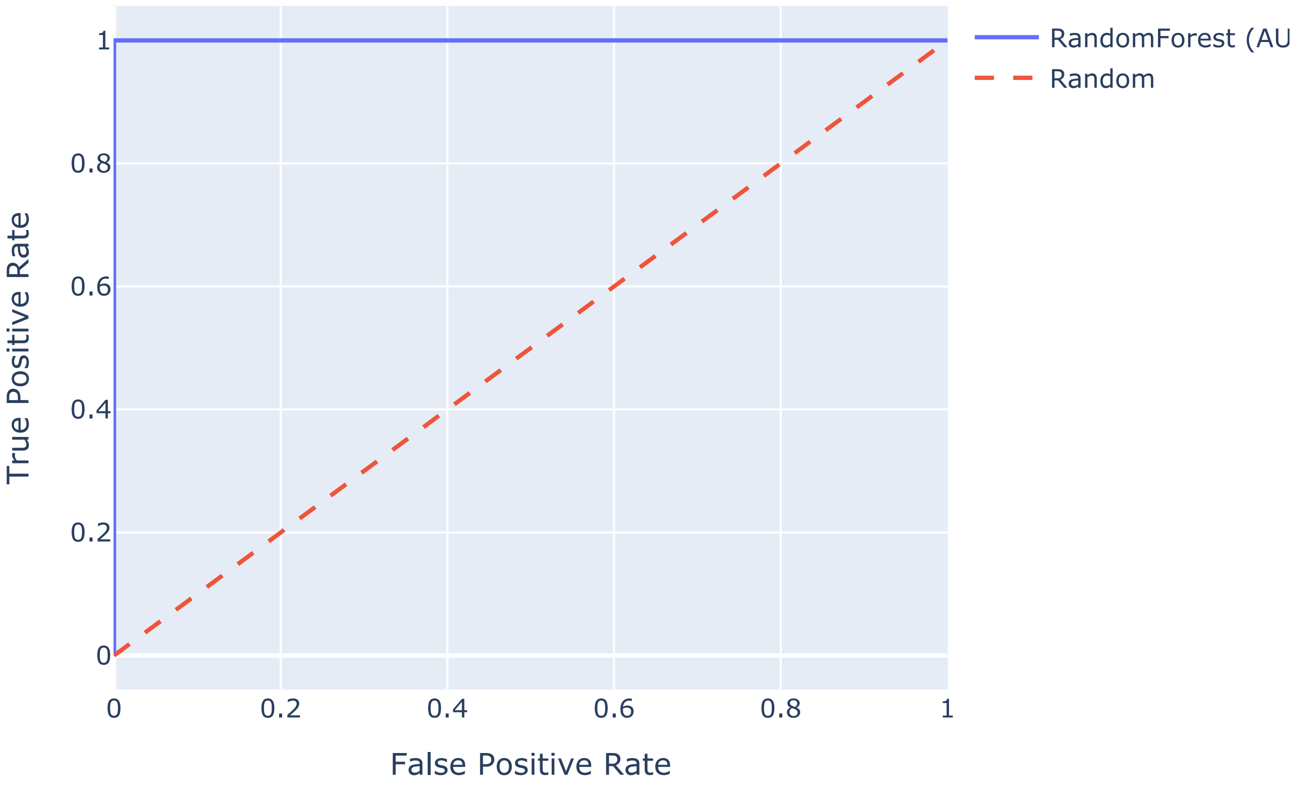
True
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
False Positive Rate

```
🔍 Target: Venom_Allergy | Model: SVM
📈 Accuracy: 0.7199
🎯 F1 (0): 0.8274 | F1 (1): 0.2312
📊 Precision: 0.8860 | AUC: 0.7180026455026456
📊 Confusion Matrix:
 [[1050    0]
 [  89    0]]
```

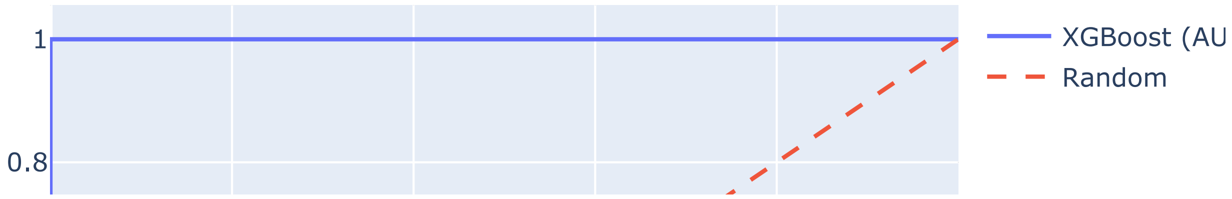


```
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedKFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.metrics import (
    f1_score, accuracy_score, recall_score,
    precision_score, confusion_matrix, roc_auc_score, roc_curve
)
from imblearn.over_sampling import SMOTE
import plotly.graph_objects as go

ALEX_sev = ALEX[ALEX["Allergy_Present"] == 1]
targets = ["Severe_Allergy"]
models = {
    "RandomForest": RandomForestClassifier(random_state=42),
    "XGBoost": XGBClassifier(random_state=42, eval_metric="logloss", use_label_
    "LogisticRegression": LogisticRegression(max_iter=1000, random_state=42),
    "SVM": SVC(probability=True, random_state=42)
}
X = ALEX_sev.copy()
X.drop(target_1, axis=1, inplace=True)
X.drop(extra_columns, axis=1, inplace=True)
X.drop(extra, axis=1, inplace=True)
X.drop(inconnu, axis=1, inplace=True)
X = X.iloc[:, 1:]
results_ALEX = []
kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

for target in targets:
    y = ALEX_sev[target]

    for model_name, base_model in models.items():
        f1_class0_scores, f1_class1_scores = [], []
        precision_scores, acc_scores, recall_scores, auc_scores = [], [], [], [

        for train_idx, test_idx in kfold.split(X, y):
            X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
            y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]

            smote = SMOTE(random_state=42)
            X_train_res, y_train_res = smote.fit_resample(X_train, y_train)

            base_model.fit(X_train_res, y_train_res)
            y_pred = base_model.predict(X_test)

            acc_scores.append(accuracy_score(y_test, y_pred))
            recall_scores.append(recall_score(y_test, y_pred, zero_division=0))
            precision_scores.append(precision_score(y_test, y_pred, average='we
            f1_class0_scores.append(f1_score(y_test, y_pred, pos_label=0, zero_
            f1_class1_scores.append(f1_score(y_test, y_pred, pos_label=1, zero_

            if hasattr(base_model, "predict_proba"):
                y_proba = base_model.predict_proba(X_test)[:, 1]
                auc_scores.append(roc_auc_score(y_test, y_proba))

        base_model.fit(X, y)
        y_pred_full = base_model.predict(X)
        y_proba_full = base_model.predict_proba(X)[:, 1] if hasattr(base_model,
        matrix = confusion_matrix(y, y_pred_full)

        print(f"\n🔍 Target: {target} | Model: {model_name}")
        print(f"📈 Accuracy: {np.mean(acc_scores):.4f}")
        print(f"🎯 F1 (0): {np.mean(f1_class0_scores):.4f} | F1 (1): {np.mean(
        print(f"📊 Precision: {np.mean(precision_scores):.4f} | AUC: {np.mean(a
        print("📊 Confusion Matrix:\n", matrix)

        if y_proba_full is not None:
            fpr, tpr, _ = roc_curve(y, y_proba_full)
            fig = go.Figure()
            fig.add_trace(go.Scatter(x=fpr, y=tpr, mode='lines', name=f"{model_
            fig.add_trace(go.Scatter(x=[0, 1], y=[0, 1], mode='lines', name='Ra
            fig.update_layout(
                title=f"ROC Curve - {target} - {model_name}",
                xaxis_title="False Positive Rate",
                yaxis_title="True Positive Rate",
                width=700, height=500
            )
            fig.show()

        results_ALEX.append({
            "Target": target,
            "Model": model_name,
            "F1_Class_0": np.mean(f1_class0_scores),
            "F1_Class_1": np.mean(f1_class1_scores),
            "Precision": np.mean(precision_scores),
            "Accuracy": np.mean(acc_scores),
            "Recall": np.mean(recall_scores),
            "AUC_ROC": np.mean(auc_scores) if auc_scores else np.nan
        })

pd.DataFrame(results_ALEX).to_csv("results_ALEX_severe.csv", index=False)
```

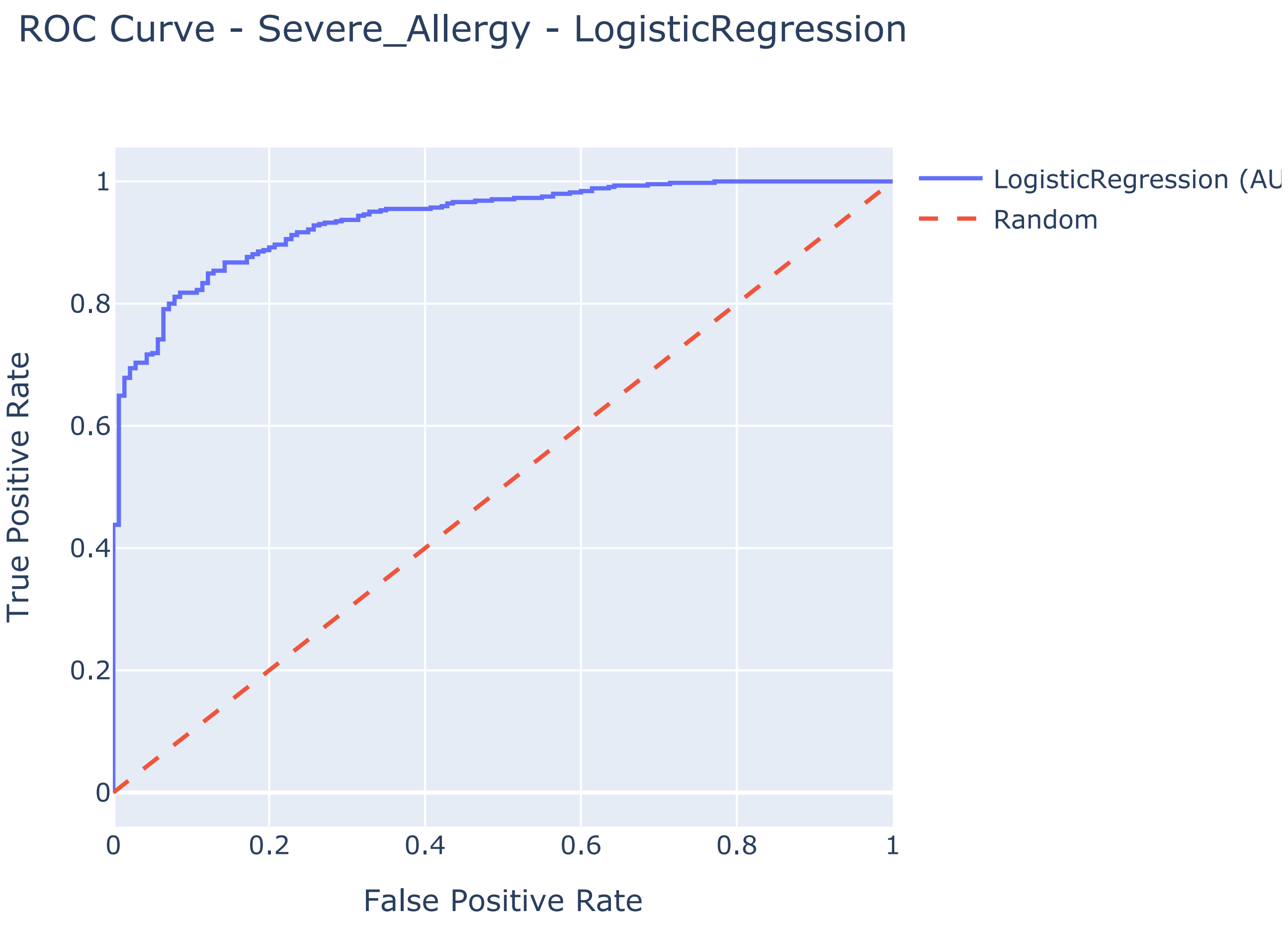
```
🔍 Target: Severe_Allergy | Model: RandomForest
📈 Accuracy: 0.8360
📈 Accuracy: 0.8360
🎯 F1 (0): 0.6370 | F1 (1): 0.8936
📊 Precision: 0.8368 | AUC: 0.8860155122655122
📊 Confusion Matrix:
 [[140   0]
 [  0 445]]
```


ROC Curve - Severe_Allergy - RandomForest

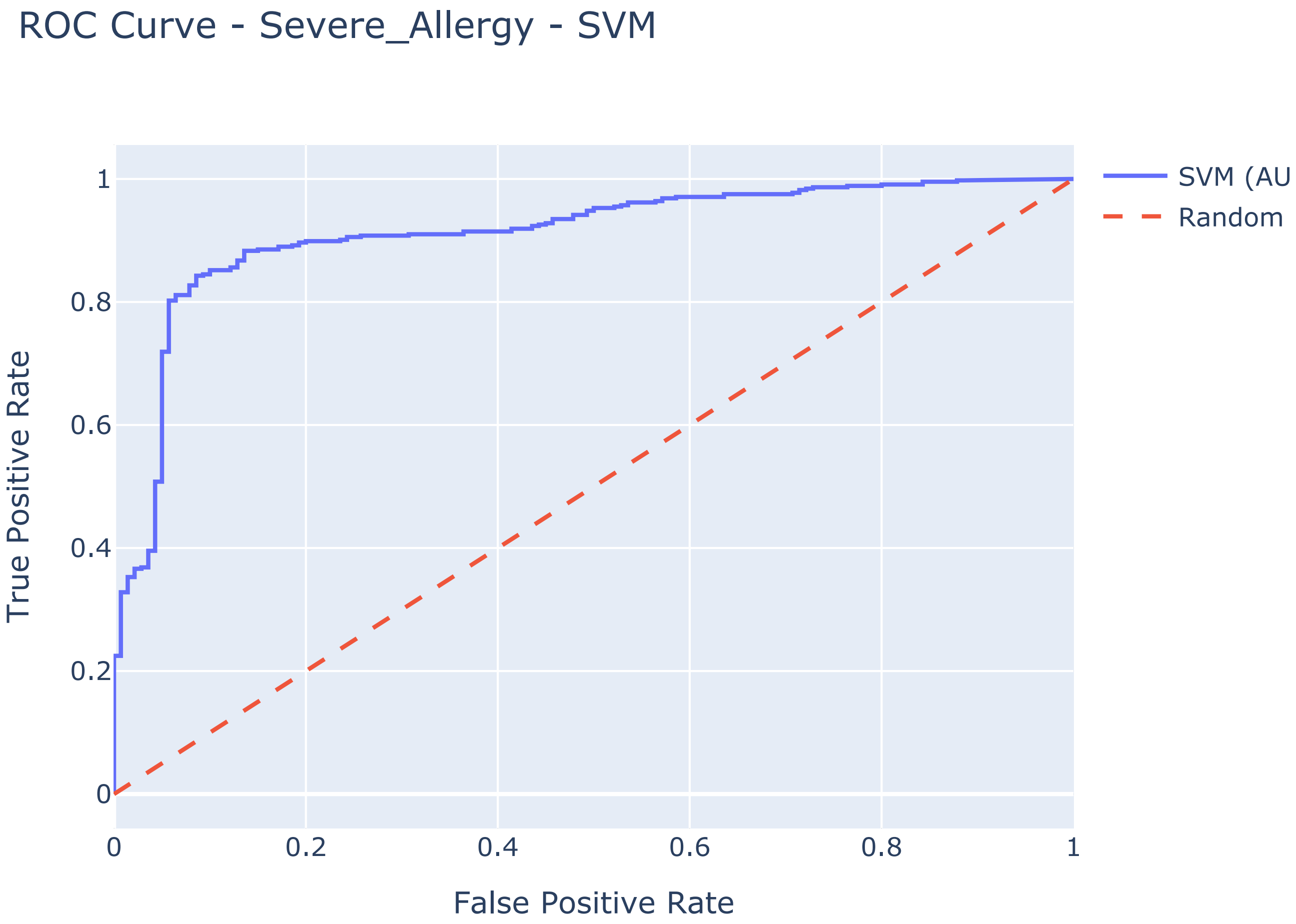
```
🔍 Target: Severe_Allergy | Model: XGBoost
📈 Accuracy: 0.8189
🎯 F1 (0): 0.6215 | F1 (1): 0.8804
📊 Precision: 0.8217 | AUC: 0.884549062049062
📊 Confusion Matrix:
 [[140   0]
 [  0 445]]
```


ROC Curve - Severe_Allergy - XGBoost

```
🔍 Target: Severe_Allergy | Model: LogisticRegression
📈 Accuracy: 0.8070
🎯 F1 (0): 0.6452 | F1 (1): 0.8667
📊 Precision: 0.8319 | AUC: 0.8610173160173161
📊 Confusion Matrix:
 [[ 96  44]
 [ 27 418]]
```


ROC Curve - Severe_Allergy - LogisticRegression

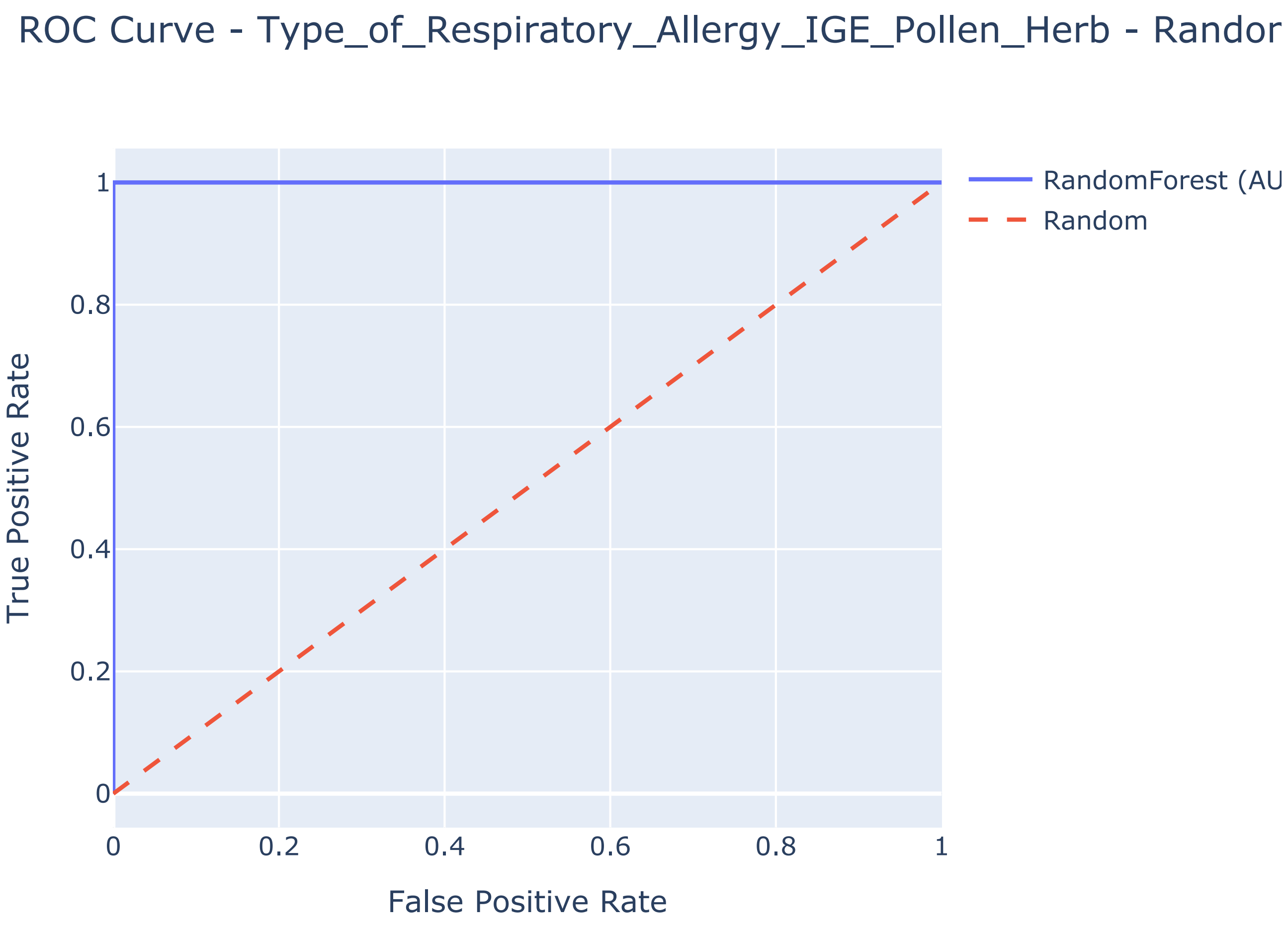
```
🔍 Target: Severe_Allergy | Model: SVM
📈 Accuracy: 0.6289
🎯 F1 (0): 0.4227 | F1 (1): 0.7232
📊 Precision: 0.7136 | AUC: 0.6493975468975469
📊 Confusion Matrix:
 [[  0 140]
 [  0 445]]
```



```
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedKFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.metrics import (
    f1_score, accuracy_score, recall_score,
    precision_score, confusion_matrix, roc_auc_score, roc_curve
)
```

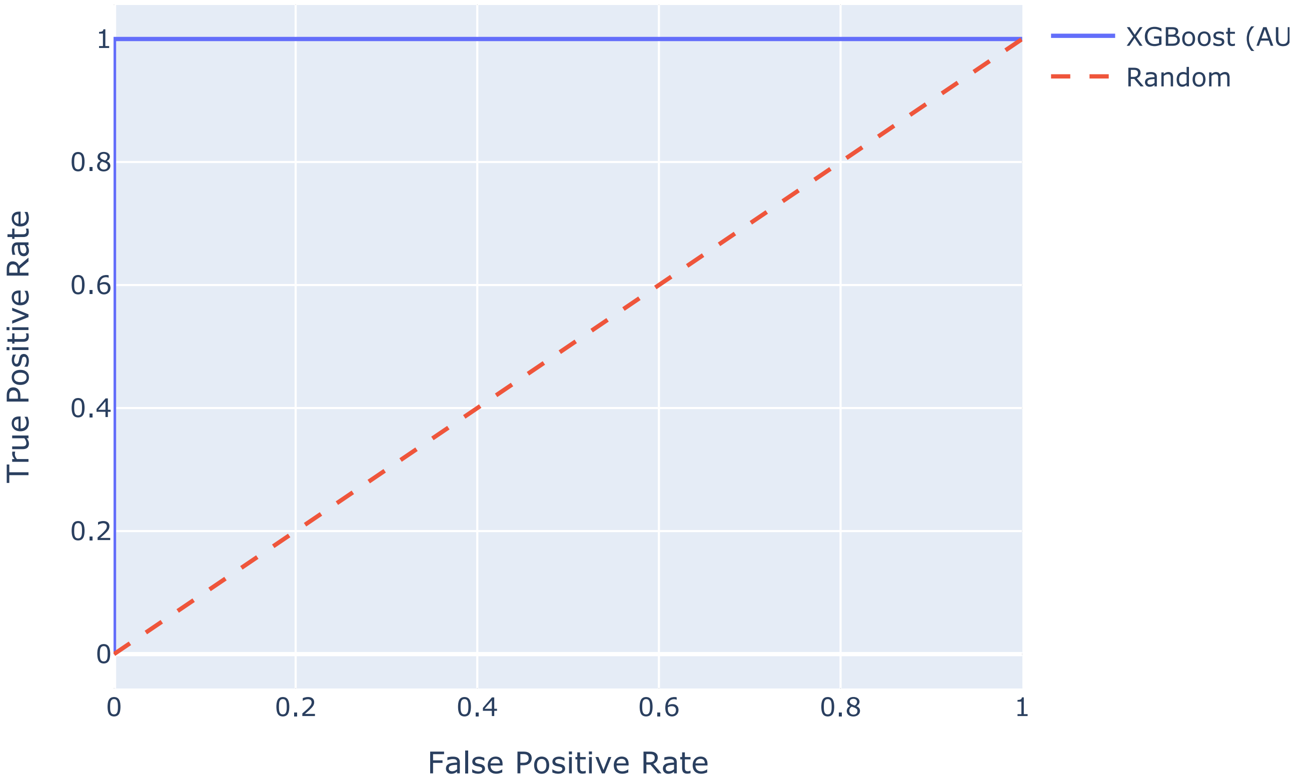
```
from imblearn.over_sampling import SMOTE
import plotly.graph_objects as go

# Données respiratoires
ALEX_res = ALEX[ALEX["Respiratory_Allergy"] == 1]

targets = ["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"]

models = {
    "RandomForest": RandomForestClassifier(random_state=42),
    "XGBoost": XGBClassifier(random_state=42, eval_metric="logloss", use_label_
    "LogisticRegression": LogisticRegression(max_iter=1000, random_state=42),
    "SVM": SVC(probability=True, random_state=42)
}

X = ALEX_res.copy()
X.drop(target_1, axis=1, inplace=True)
X.drop(extra_columns, axis=1, inplace=True)
X.drop(extra, axis=1, inplace=True)
X.drop(inconnu, axis=1, inplace=True)
X = X.iloc[:, 1:]

results_ALEX_res = []
kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

# Boucle principale
for target in targets:
    y = ALEX_res[target]

    for model_name, base_model in models.items():
        f1_class0_scores, f1_class1_scores = [], []
        precision_scores, acc_scores, recall_scores, auc_scores = [], [], [], [

        print(f"\n🔍 Target: {target} | Model: {model_name}")

        for train_idx, test_idx in kfold.split(X, y):
            X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
            y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]

            # Application de SMOTE sur les données d'entraînement
            smote = SMOTE(random_state=42)
            X_train_res, y_train_res = smote.fit_resample(X_train, y_train)

            base_model.fit(X_train_res, y_train_res)
            y_pred = base_model.predict(X_test)

            acc_scores.append(accuracy_score(y_test, y_pred))
            recall_scores.append(recall_score(y_test, y_pred, zero_division=0))
            precision_scores.append(precision_score(y_test, y_pred, average='we
            f1_class0_scores.append(f1_score(y_test, y_pred, pos_label=0, zero_
            f1_class1_scores.append(f1_score(y_test, y_pred, pos_label=1, zero_

            if hasattr(base_model, "predict_proba"):
                y_proba = base_model.predict_proba(X_test)[:, 1]
                auc_scores.append(roc_auc_score(y_test, y_proba))

        # Entraînement final sur tout X (sans SMOTE ici, car prédiction globale
        base_model.fit(X, y)
        y_pred_full = base_model.predict(X)
        y_proba_full = base_model.predict_proba(X)[:, 1] if hasattr(base_model,
        matrix = confusion_matrix(y, y_pred_full)

        print(f"📈 Accuracy: {np.mean(acc_scores):.4f}")
        print(f"🎯 F1 (0): {np.mean(f1_class0_scores):.4f} | F1 (1): {np.mean(
        print(f"📊 Precision: {np.mean(precision_scores):.4f} | AUC: {np.mean(a
        print("📊 Confusion Matrix:\n", matrix)

        if y_proba_full is not None:
            fpr, tpr, _ = roc_curve(y, y_proba_full)
            fig = go.Figure()
            fig.add_trace(go.Scatter(x=fpr, y=tpr, mode='lines', name=f"{model_
            fig.add_trace(go.Scatter(x=[0, 1], y=[0, 1], mode='lines', name='Ra
            fig.update_layout(
                title=f"ROC Curve - {target} - {model_name}",
                xaxis_title="False Positive Rate",
                yaxis_title="True Positive Rate",
                width=700, height=500
            )
            fig.show()

        results_ALEX_res.append({
            "Target": target,
            "Model": model_name,
            "F1_Class_0": np.mean(f1_class0_scores),
            "F1_Class_1": np.mean(f1_class1_scores),
            "Precision": np.mean(precision_scores),
            "Accuracy": np.mean(acc_scores),
            "Recall": np.mean(recall_scores),
            "AUC_ROC": np.mean(auc_scores) if auc_scores else np.nan
        })

pd.DataFrame(results_ALEX_res).to_csv("results_ALEX_respiratoire.csv", index=Fa
```

```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Pollen_Herb | Model: RandomFores
📈 Accuracy: 0.7533
🎯 F1 (0): 0.7527 | F1 (1): 0.7520
📊 Precision: 0.7565 | AUC: 0.8329328205128206
📊 Confusion Matrix:
 [[252   0]
 [  0 243]]
```



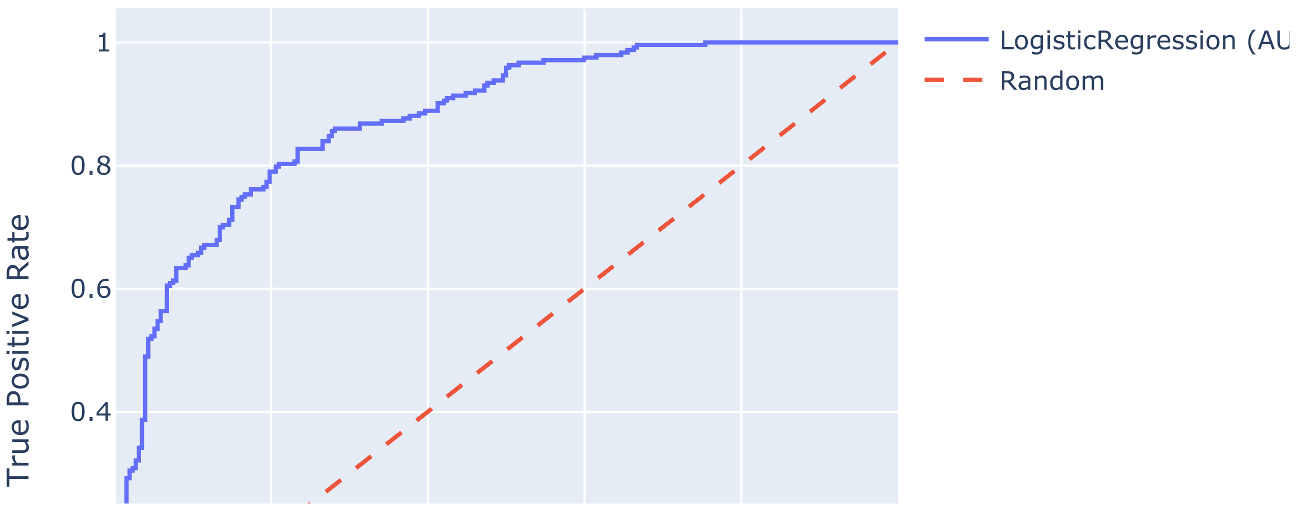
```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Pollen_Herb | Model: XGBoost
📈 Accuracy: 0.7571
🎯 F1 (0): 0.7542 | F1 (1): 0.7583
📊 Precision: 0.7619 | AUC: 0.823191282051282
📊 Confusion Matrix:
 [[252   0]
 [  0 243]]
```

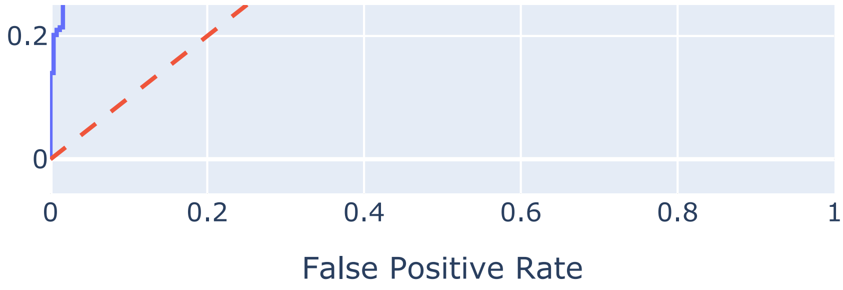
ROC Curve - Type_of_Respiratory_Allergy_IGE_Pollen_Herb - XGBoos



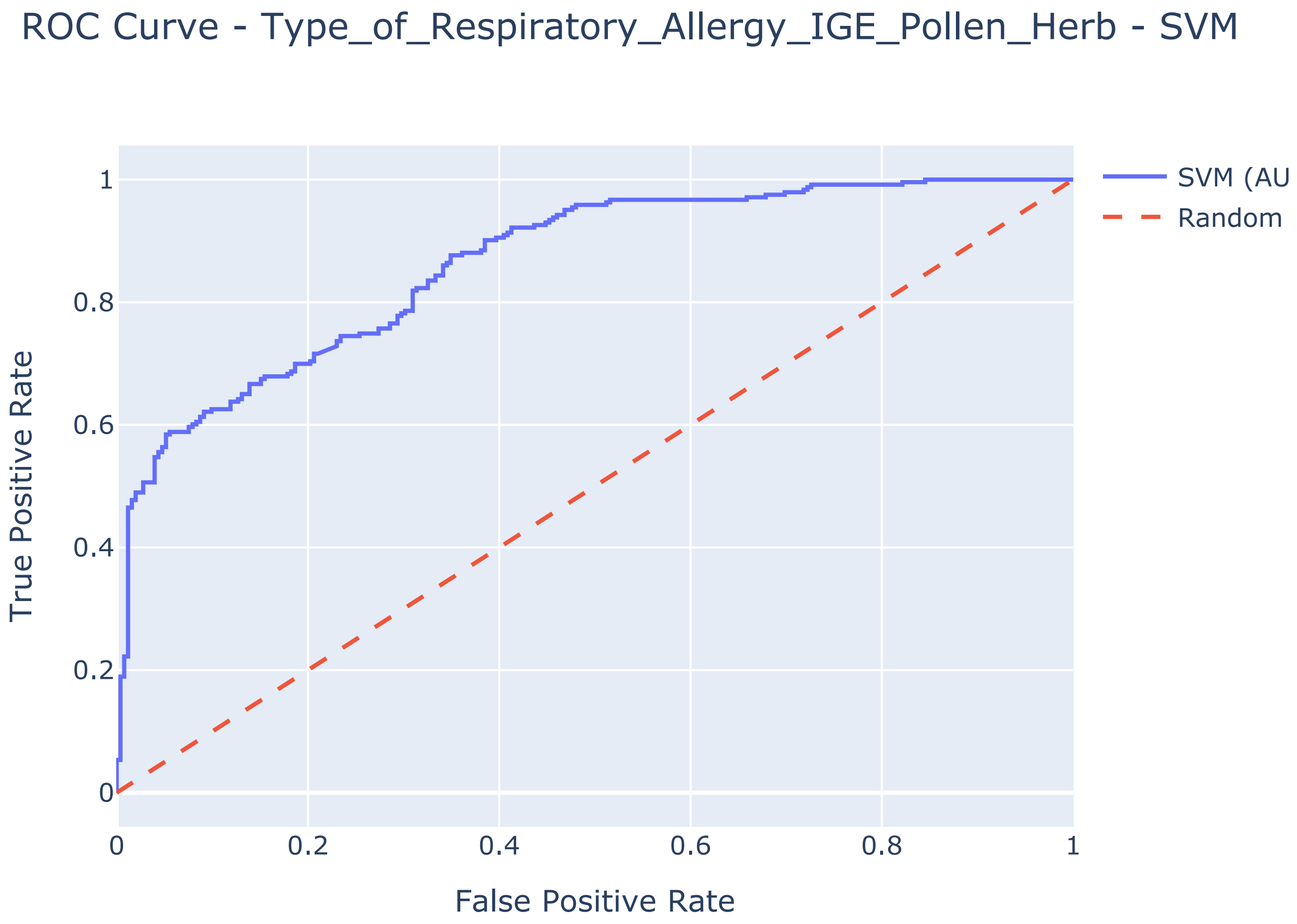
```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Pollen_Herb | Model: LogisticReg
📈 Accuracy: 0.7009
🎯 F1 (0): 0.7151 | F1 (1): 0.6838
📊 Precision: 0.7043 | AUC: 0.7811423076923077
📊 Confusion Matrix:
 [[210  42]
 [ 60 183]]
```

ROC Curve - Type_of_Respiratory_Allergy_IGE_Pollen_Herb - Logistic



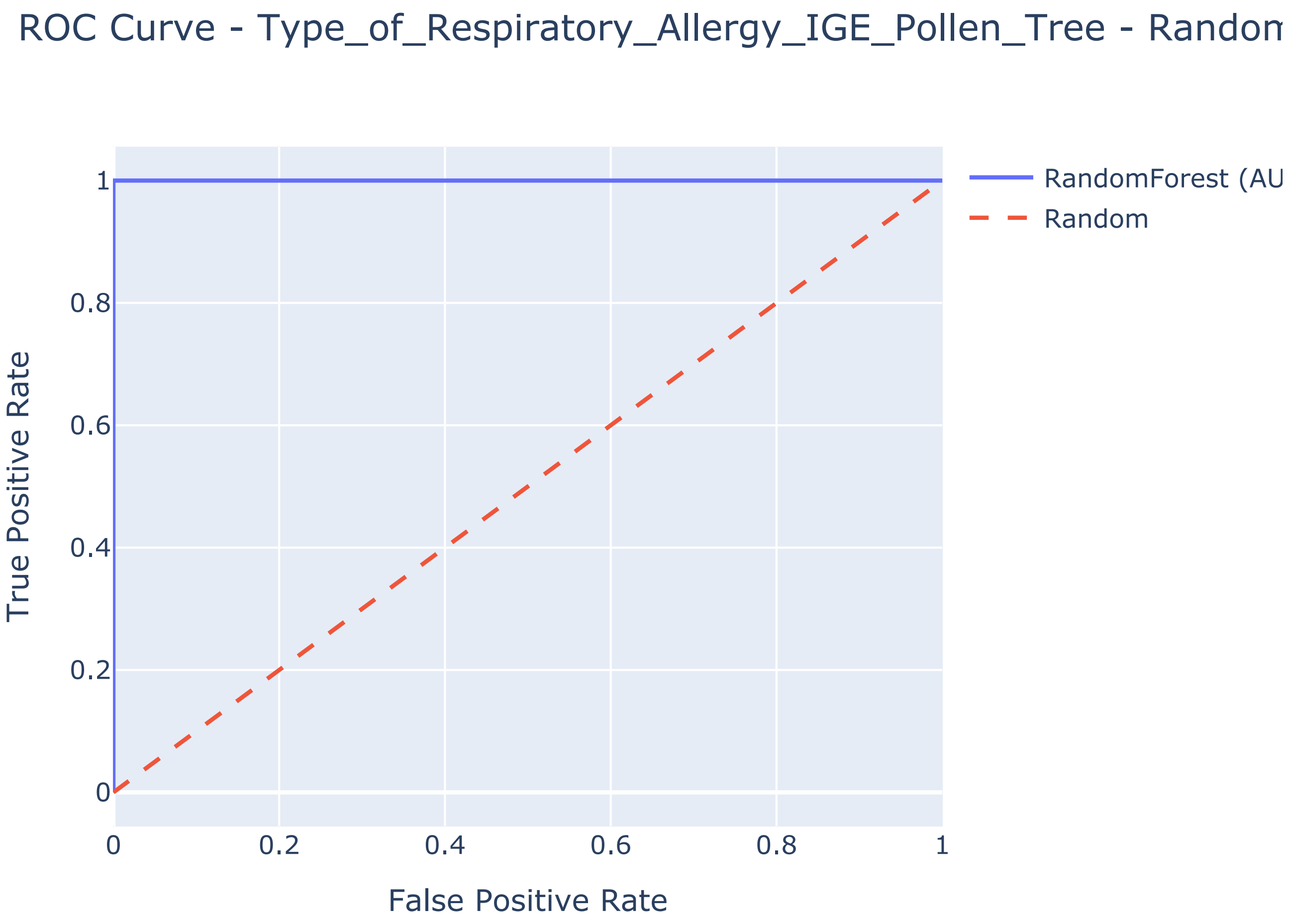


```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Pollen_Herb | Model: SVM
📈 Accuracy: 0.6991
🎯 F1 (0): 0.7099 | F1 (1): 0.6856
📊 Precision: 0.7028 | AUC: 0.802943846153846
📊 Confusion Matrix:
 [[200  52]
 [ 71 172]]
```

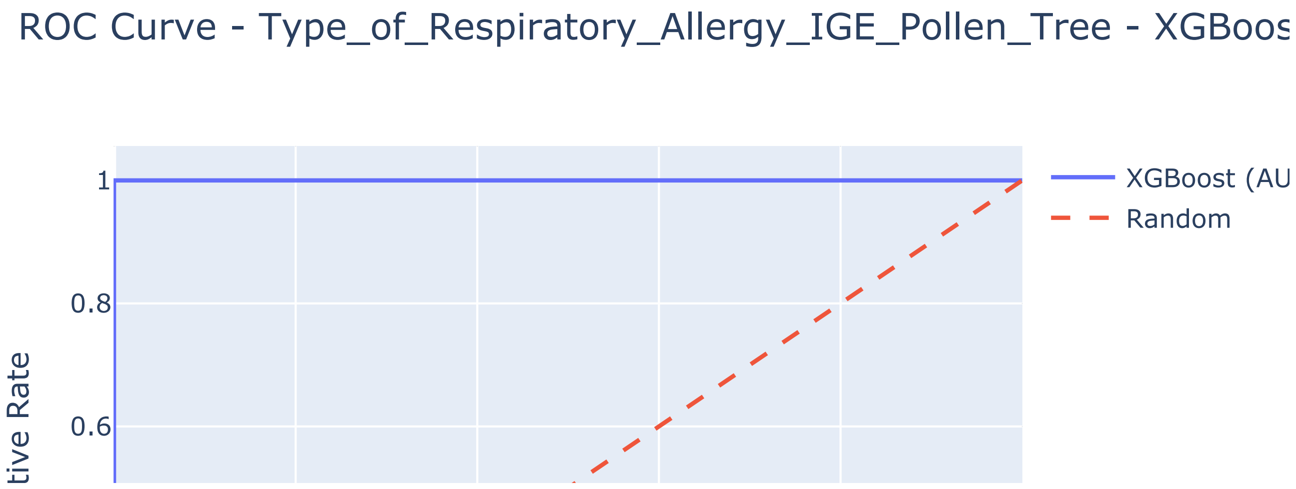


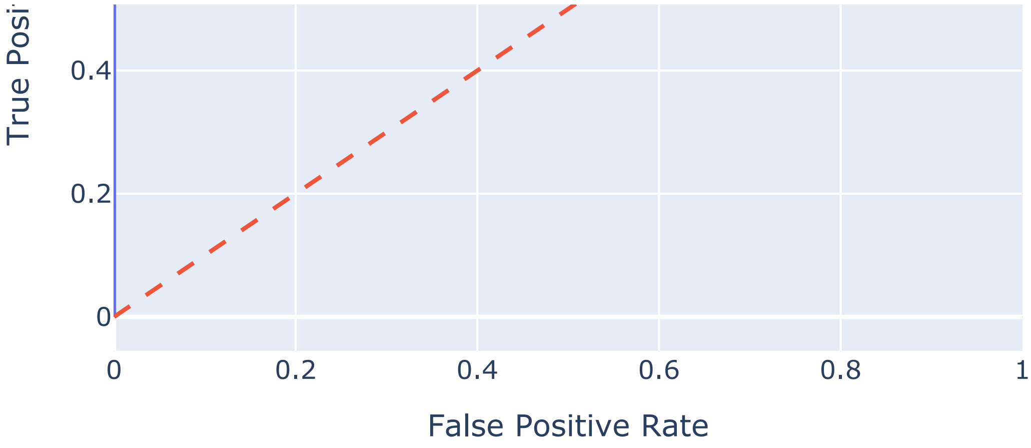
```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Pollen_Tree | Model: RandomFores
📈 Accuracy: 0.8687
🎯 F1 (0): 0.8227 | F1 (1): 0.8951
📊 Precision: 0.8725 | AUC: 0.9461661125259384
📊 Confusion Matrix:
 [[182    0]
```

```
[[182    0]
 [   0  313]]
```

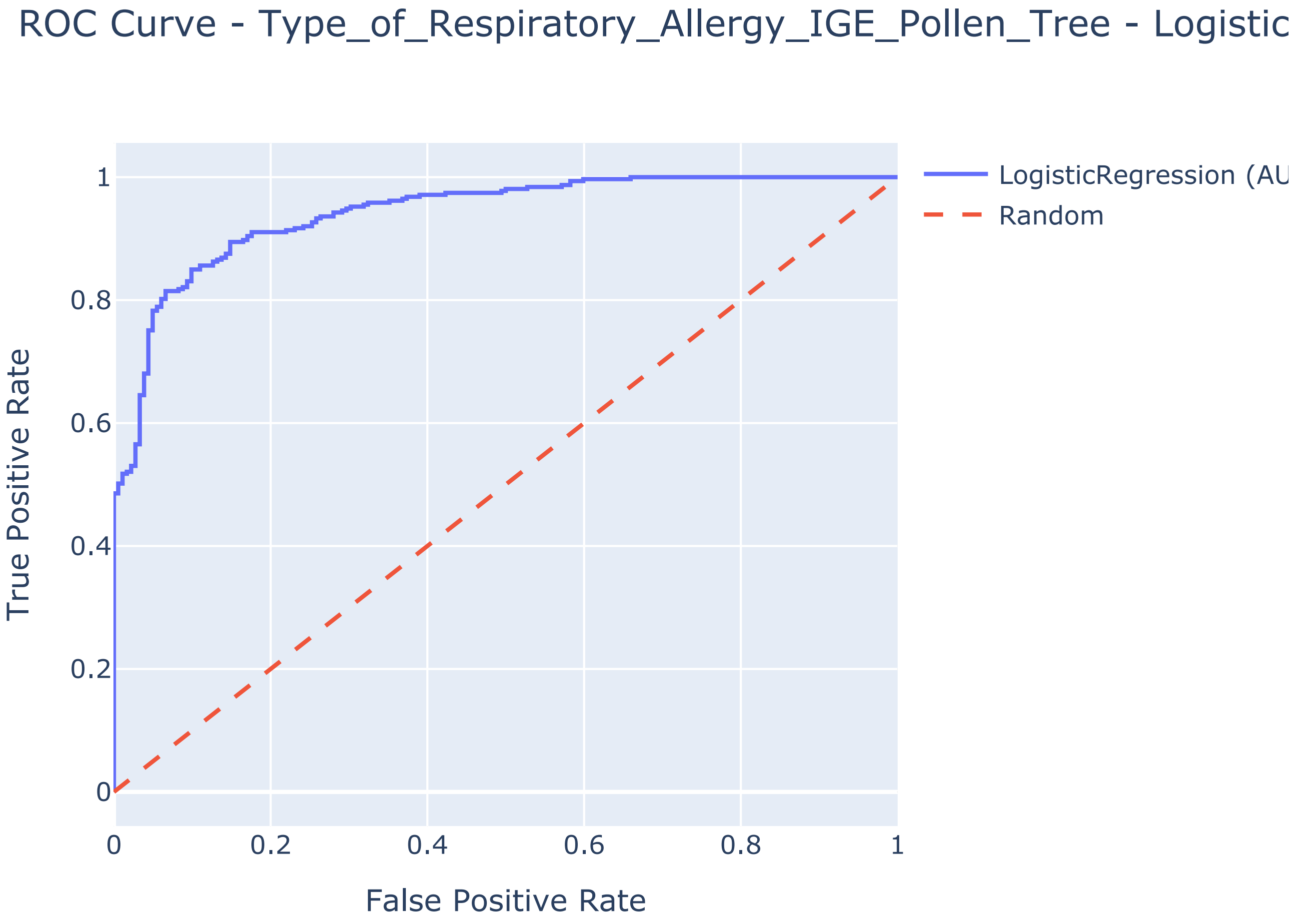


```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Pollen_Tree | Model: XGBoost
📈 Accuracy: 0.8849
🎯 F1 (0): 0.8445 | F1 (1): 0.9083
📊 Precision: 0.8867 | AUC: 0.9509175745142425
📊 Confusion Matrix:
[[182    0]
 [   0  313]]
```



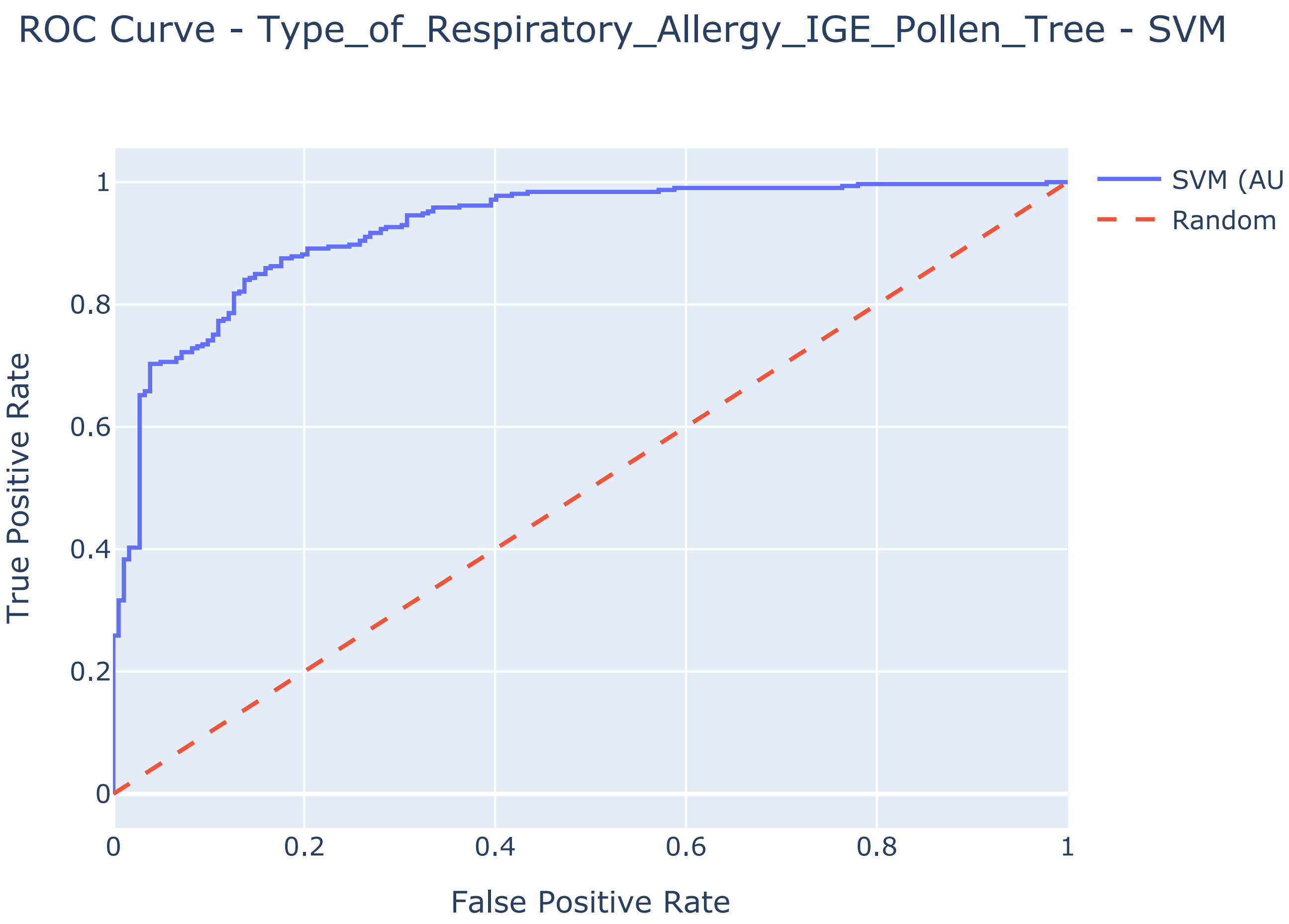


```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Pollen_Tree | Model: LogisticReg
📈 Accuracy: 0.8099
🎯 F1 (0): 0.7545 | F1 (1): 0.8431
📊 Precision: 0.8222 | AUC: 0.8701919449160535
📊 Confusion Matrix:
 [[155  27]
 [ 33 280]]
```

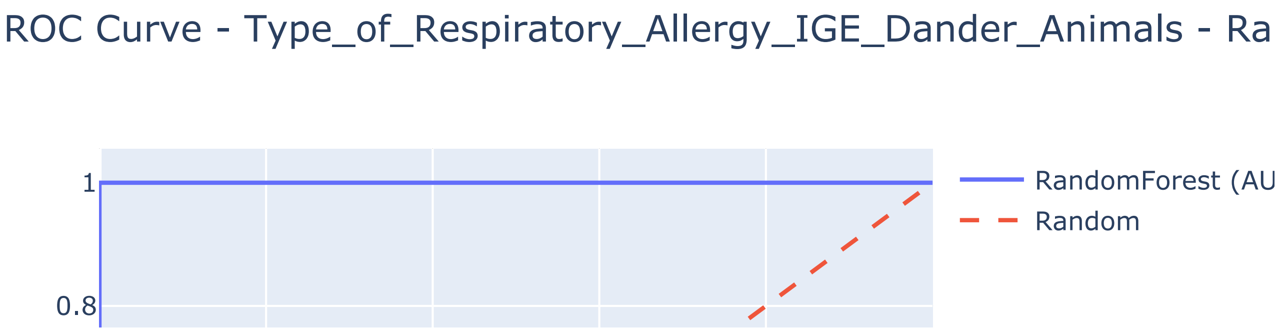


```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Pollen_Tree | Model: SVM
```

```
📈 Accuracy: 0.7733
🎯 F1 (0): 0.7310 | F1 (1): 0.8027
📊 Precision: 0.8022 | AUC: 0.8862154546312017
📊 Confusion Matrix:
 [[157  25]
 [ 50 263]]
```

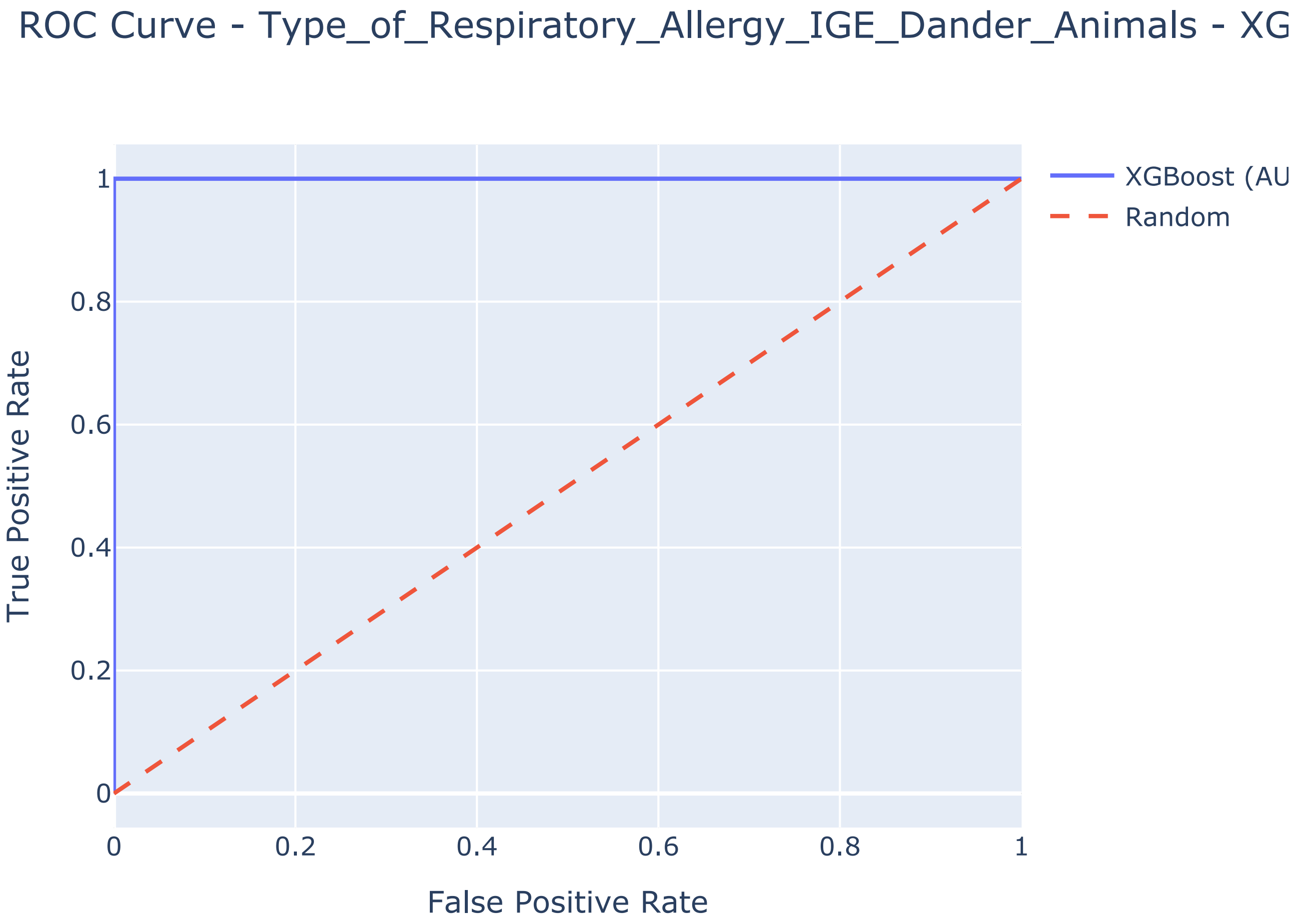


```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Dander_Animals | Model: RandomFc
📈 Accuracy: 0.8987
🎯 F1 (0): 0.8884 | F1 (1): 0.9064
📊 Precision: 0.9060 | AUC: 0.949144994988591
📊 Confusion Matrix:
 [[214   0]
 [  0 281]]
```



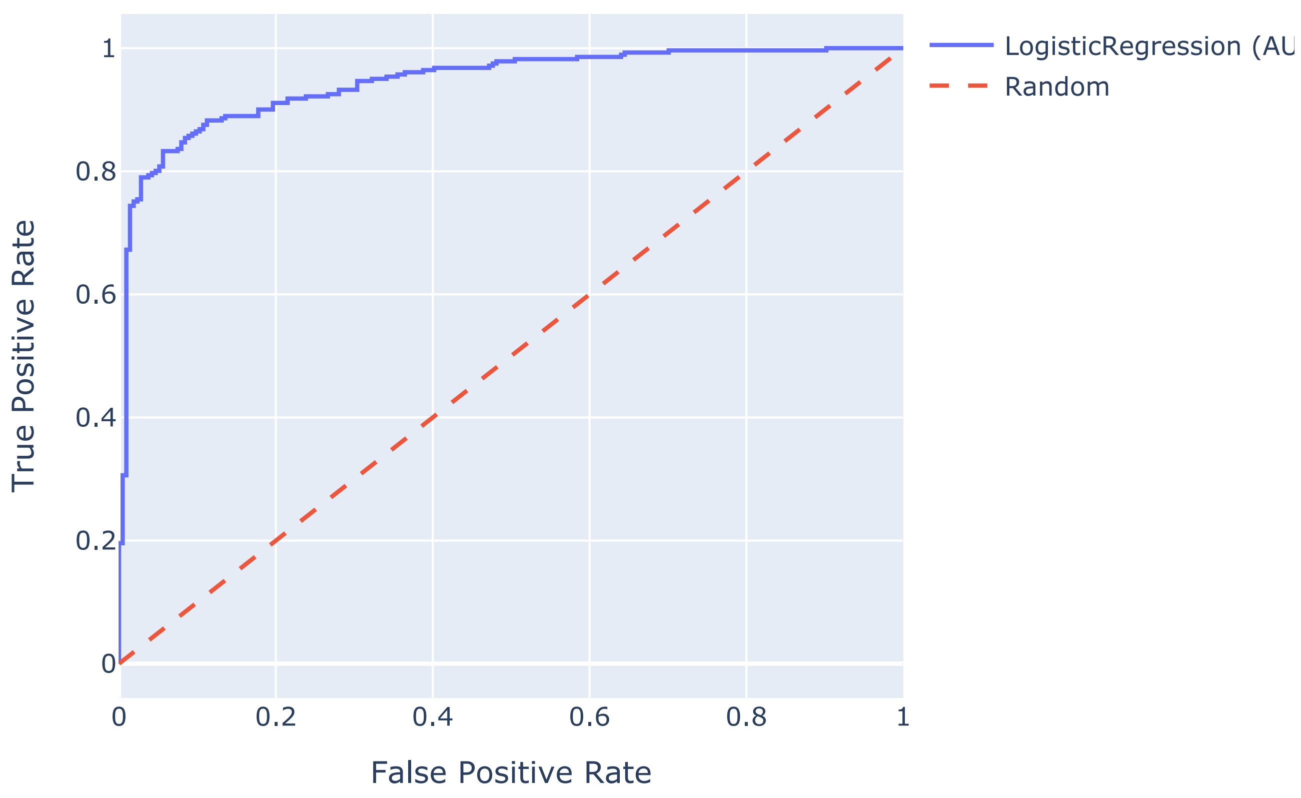
True Positive Rate
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
False Positive Rate

```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Dander_Animals | Model: XGBoost
📈 Accuracy: 0.8989
🎯 F1 (0): 0.8856 | F1 (1): 0.9088
📊 Precision: 0.9044 | AUC: 0.9427795726441047
📊 Confusion Matrix:
 [[214   0]
 [  0 281]]
```


ROC Curve - Type_of_Respiratory_Allergy_IGE_Dander_Animals - XG

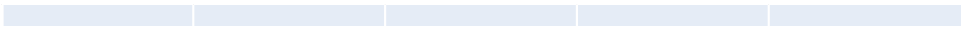
```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Dander_Animals | Model: Logistic
📈 Accuracy: 0.8179
🎯 F1 (0): 0.8056 | F1 (1): 0.8268
📊 Precision: 0.8351 | AUC: 0.8823703697780052
📊 Confusion Matrix:
 [[202  12]
 [ 48 233]]
```

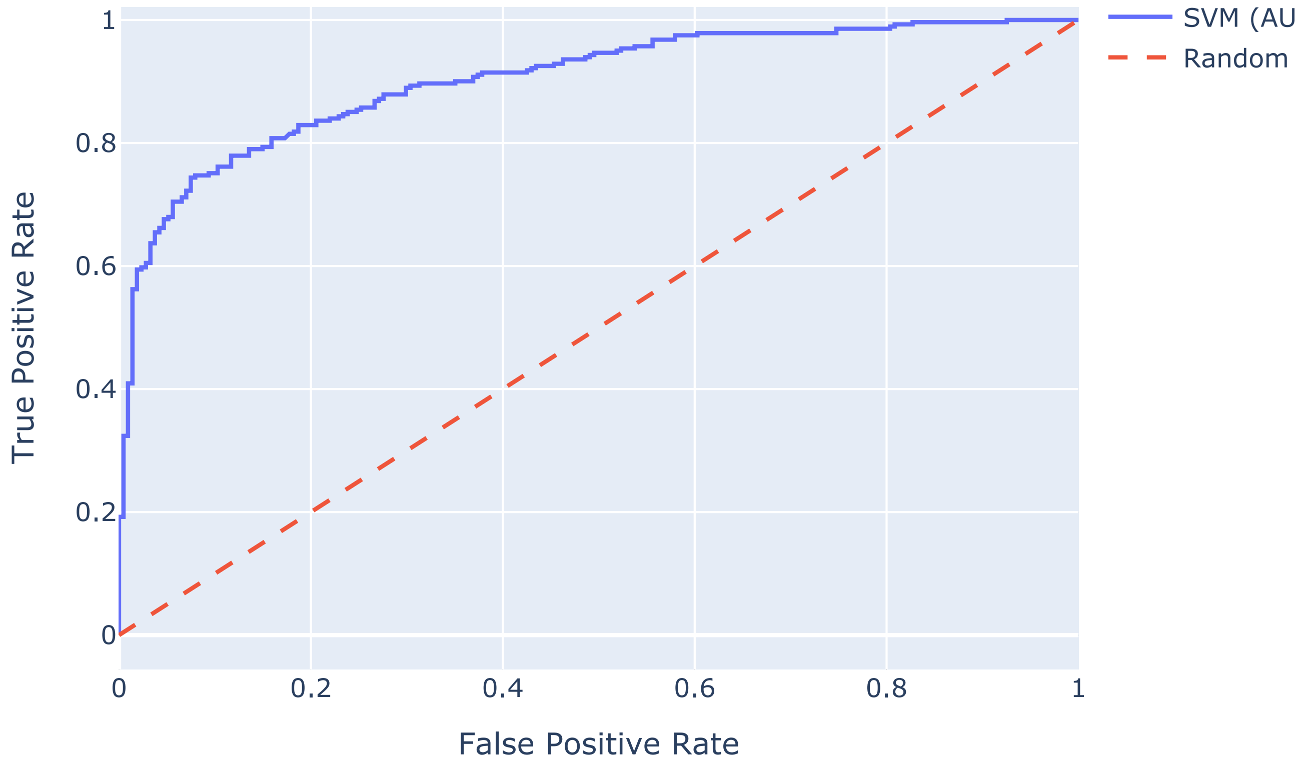
ROC Curve - Type_of_Respiratory_Allergy_IGE_Dander_Animals - Lo



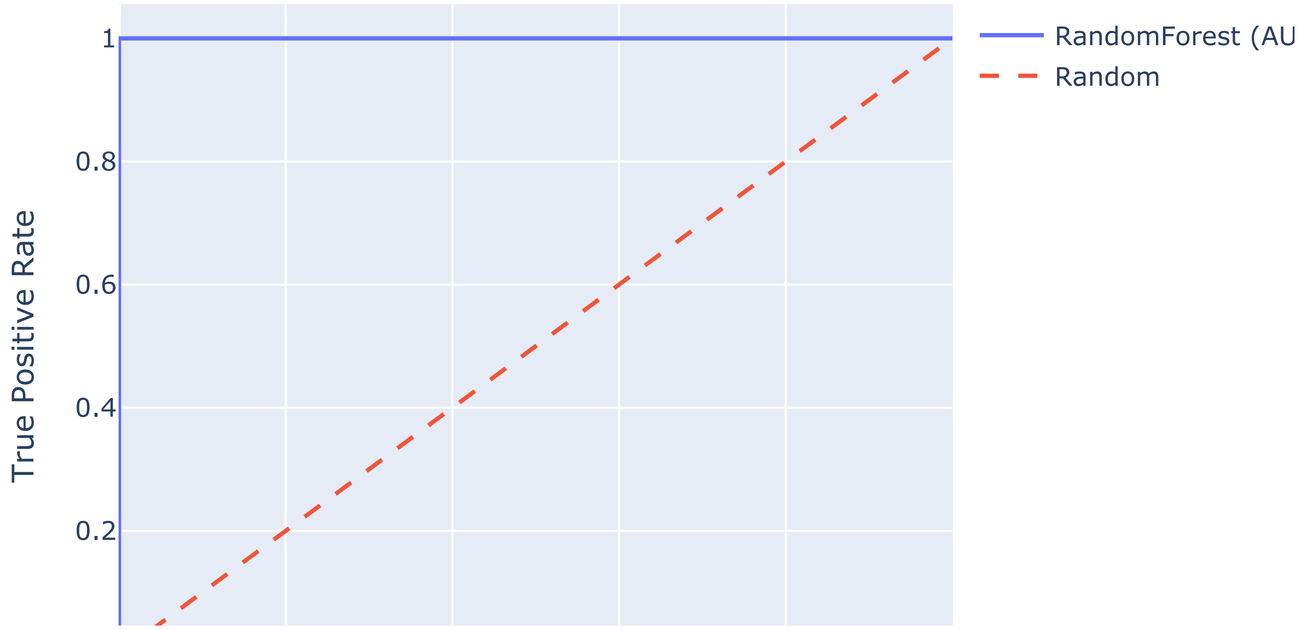
```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Dander_Animals | Model: SVM
📈 Accuracy: 0.7776
🎯 F1 (0): 0.7694 | F1 (1): 0.7808
📊 Precision: 0.8038 | AUC: 0.8713851214466978
📊 Confusion Matrix:
 [[182  32]
 [ 59 222]]
```

ROC Curve - Type_of_Respiratory_Allergy_IGE_Dander_Animals - SV

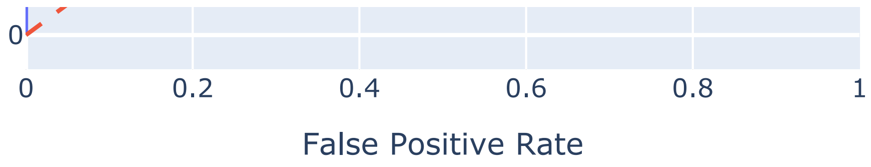




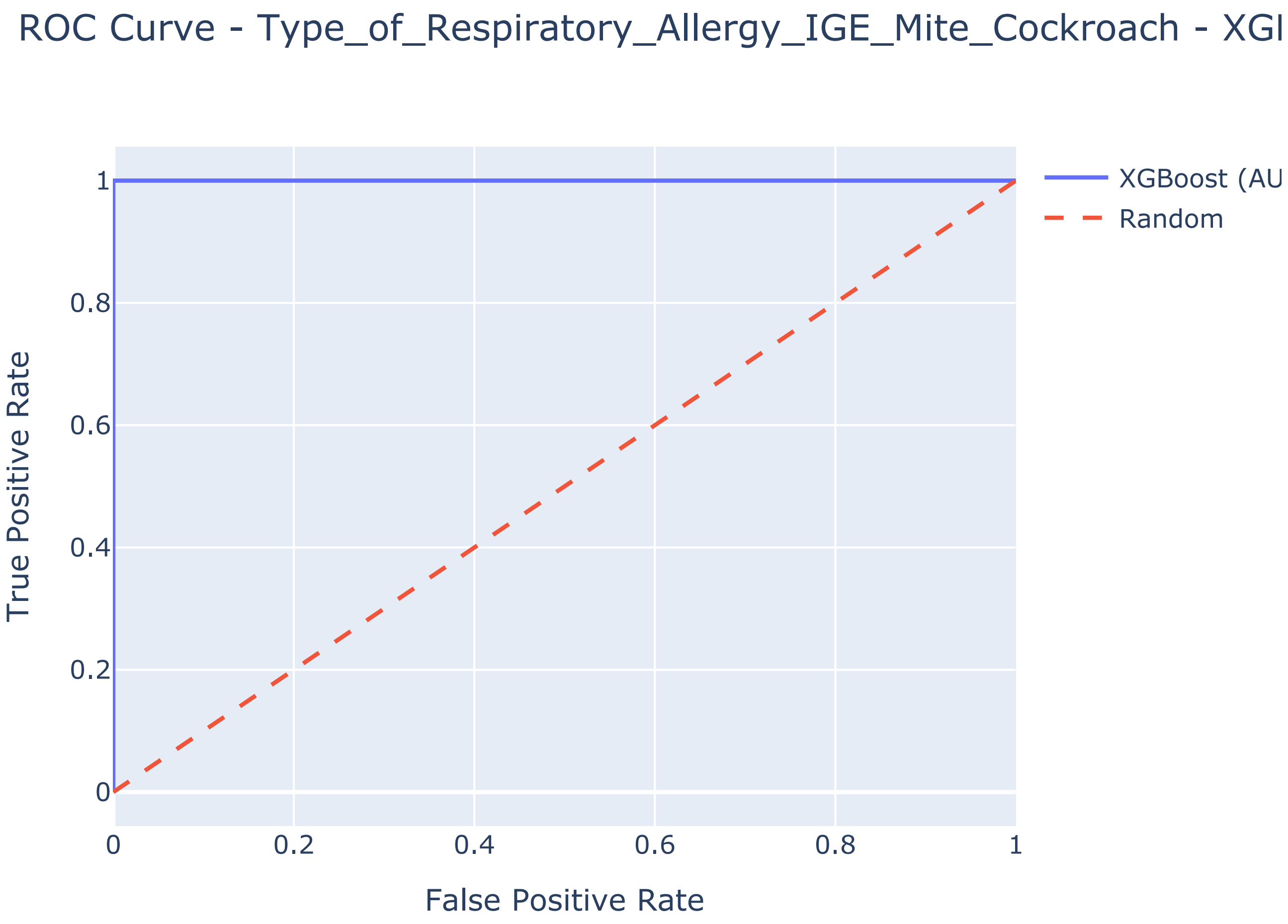
```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Mite_Cockroach | Model: RandomFc
📈 Accuracy: 0.9677
🎯 F1 (0): 0.9625 | F1 (1): 0.9715
📊 Precision: 0.9694 | AUC: 0.9907594417077178
📊 Confusion Matrix:
 [[208   0]
 [  0 287]]
```



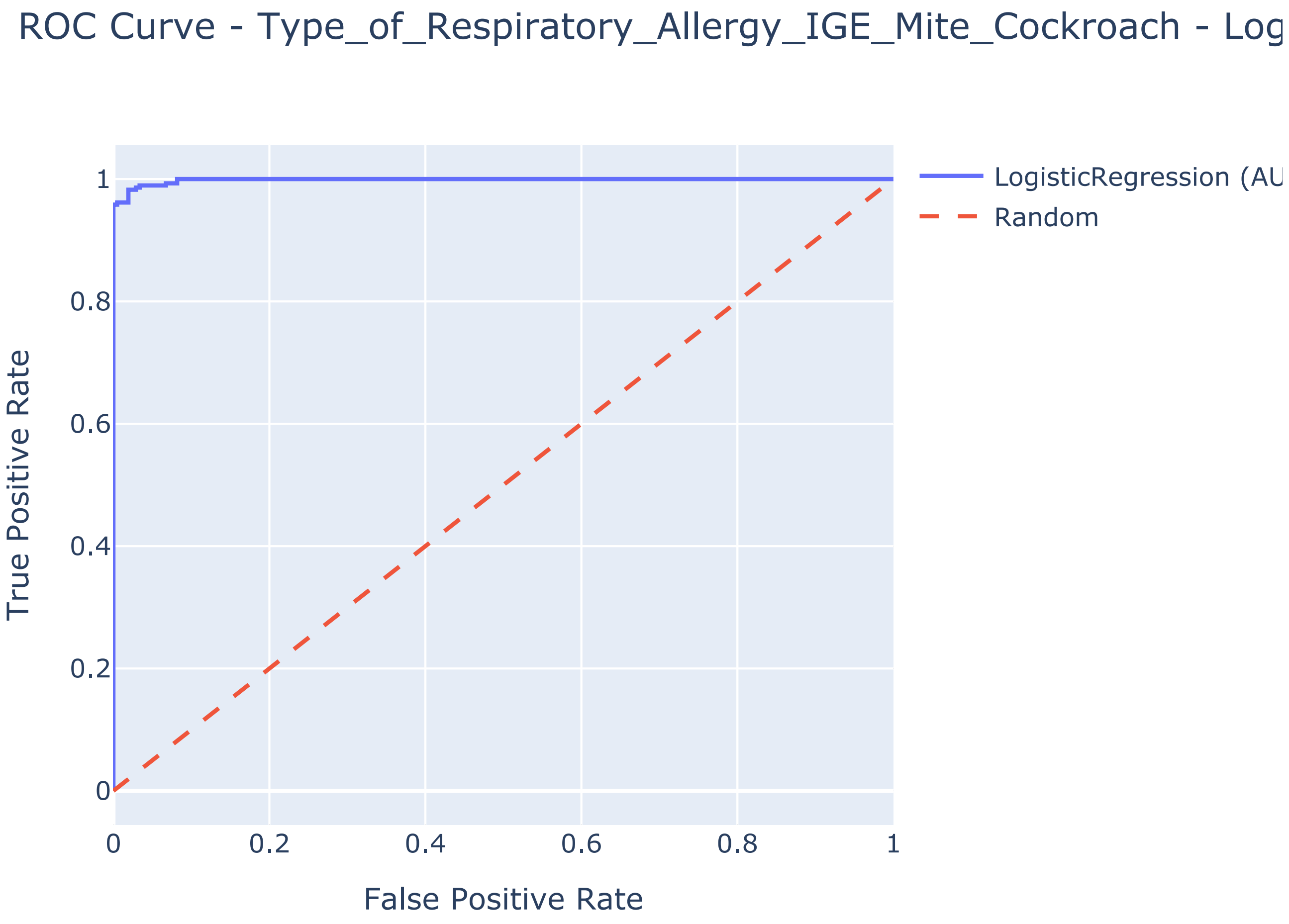
ROC Curve - Type_of_Respiratory_Allergy_IGE_Mite_Cockroach - Rar



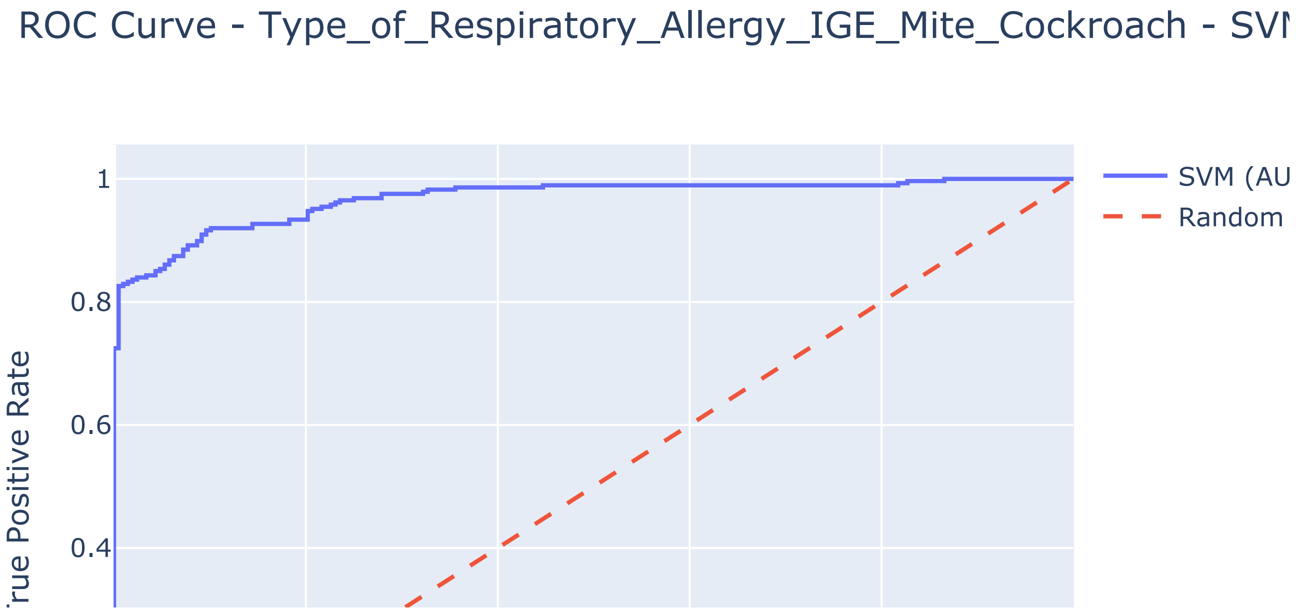
```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Mite_Cockroach | Model: XGBoost
📈 Accuracy: 0.9758
🎯 F1 (0): 0.9719 | F1 (1): 0.9786
📊 Precision: 0.9771 | AUC: 0.9879427633122215
📊 Confusion Matrix:
 [[208   0]
 [  0 287]]
```

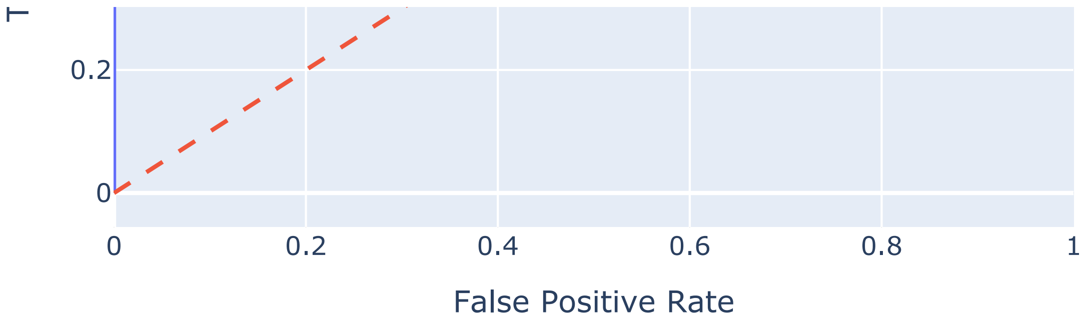


```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Mite_Cockroach | Model: Logistic
📈 Accuracy: 0.9352
🎯 F1 (0): 0.9261 | F1 (1): 0.9422
📊 Precision: 0.9393 | AUC: 0.9810755336617405
📊 Confusion Matrix:
 [[208   0]
 [ 12 275]]
```

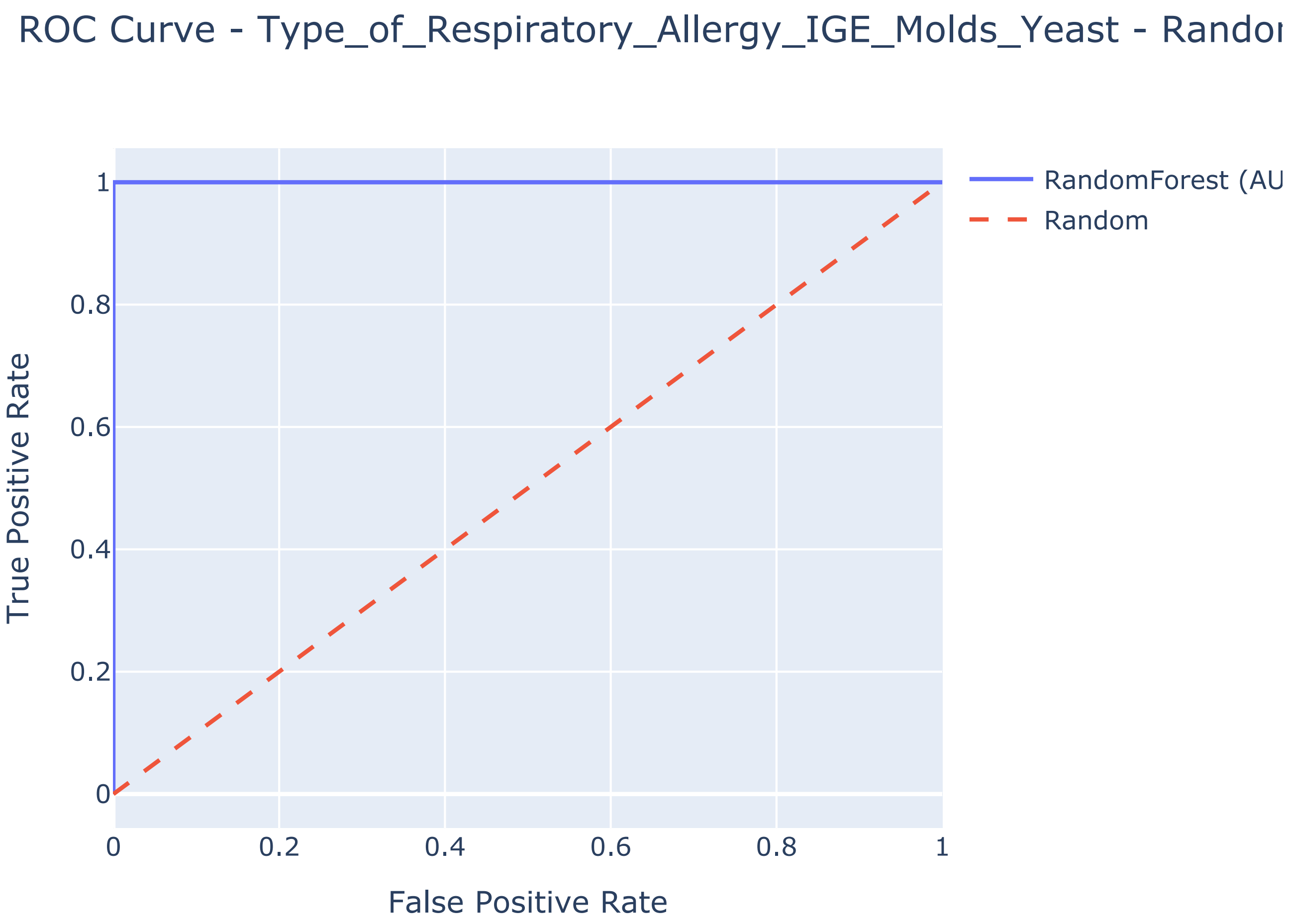


```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Mite_Cockroach | Model: SVM
📈 Accuracy: 0.8241
🎯 F1 (0): 0.8241 | F1 (1): 0.8237
📊 Precision: 0.8668 | AUC: 0.9417663617171007
📊 Confusion Matrix:
 [[207   1]
 [ 64 223]]
```





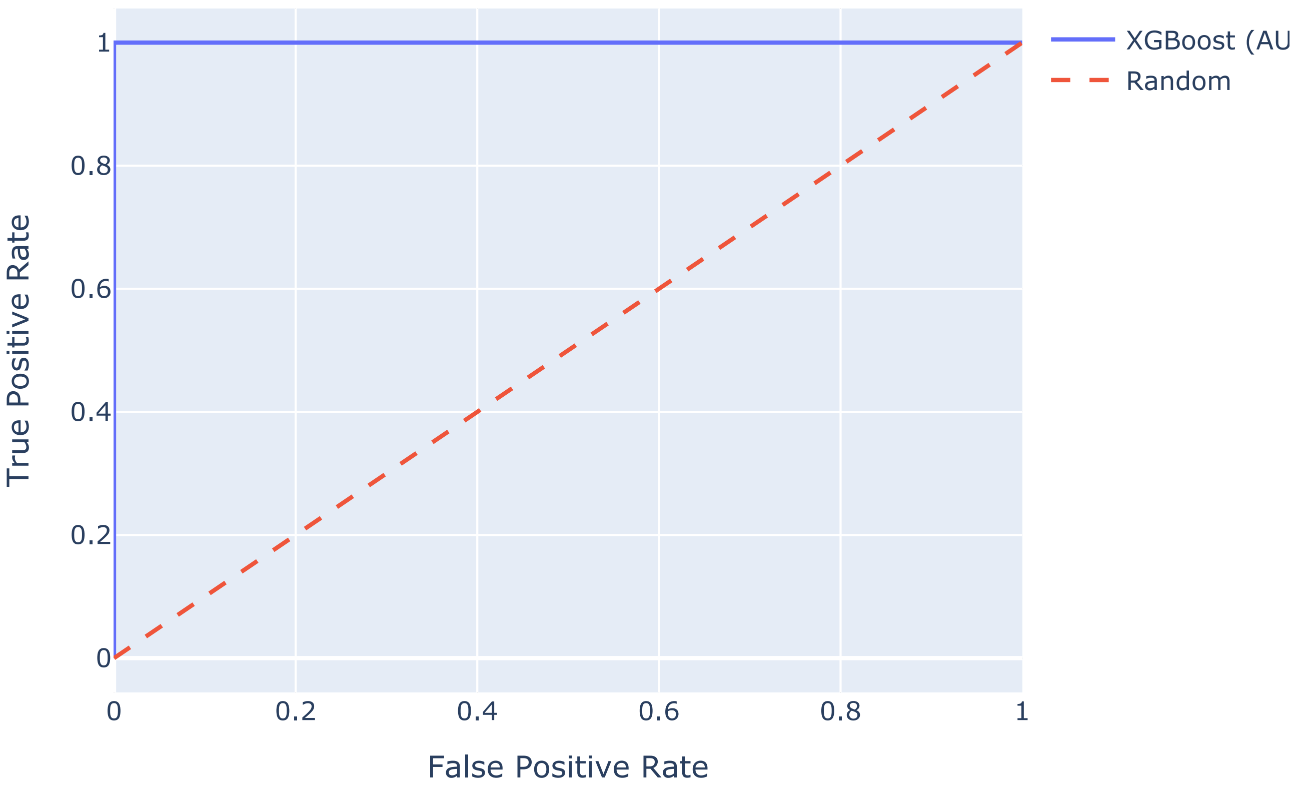
```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Molds_Yeast | Model: RandomFores
📈 Accuracy: 0.9212
🎯 F1 (0): 0.9426 | F1 (1): 0.8734
📊 Precision: 0.9258 | AUC: 0.9428471388555423
📊 Confusion Matrix:
 [[346   0]
 [  0 149]]
```


ROC Curve - Type_of_Respiratory_Allergy_IGE_Molds_Yeast - Randoı

```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Molds_Yeast | Model: XGBoost
📈 Accuracy: 0.9394
🎯 F1 (0): 0.9561 | F1 (1): 0.9020
📊 Precision: 0.9424 | AUC: 0.9498687474989996
📊 Confusion Matrix:
```

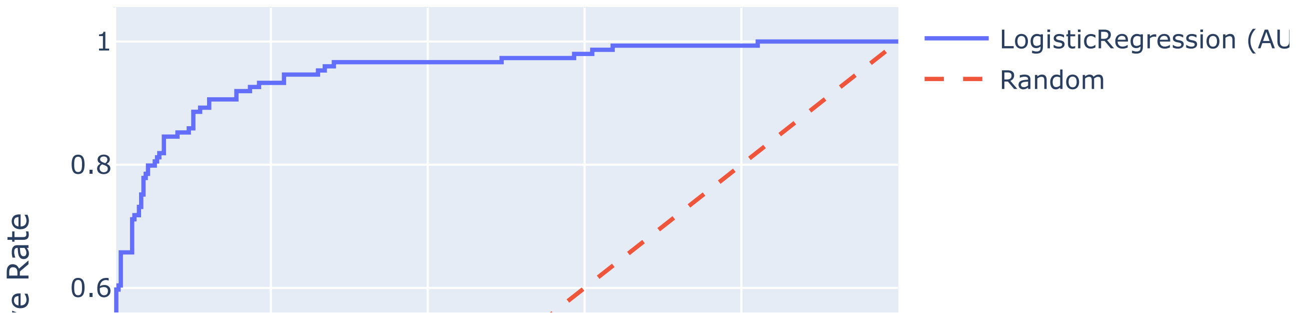
```
[[346   0]
 [  0 149]]
```

ROC Curve - Type_of_Respiratory_Allergy_IGE_Molds_Yeast - XGBoo

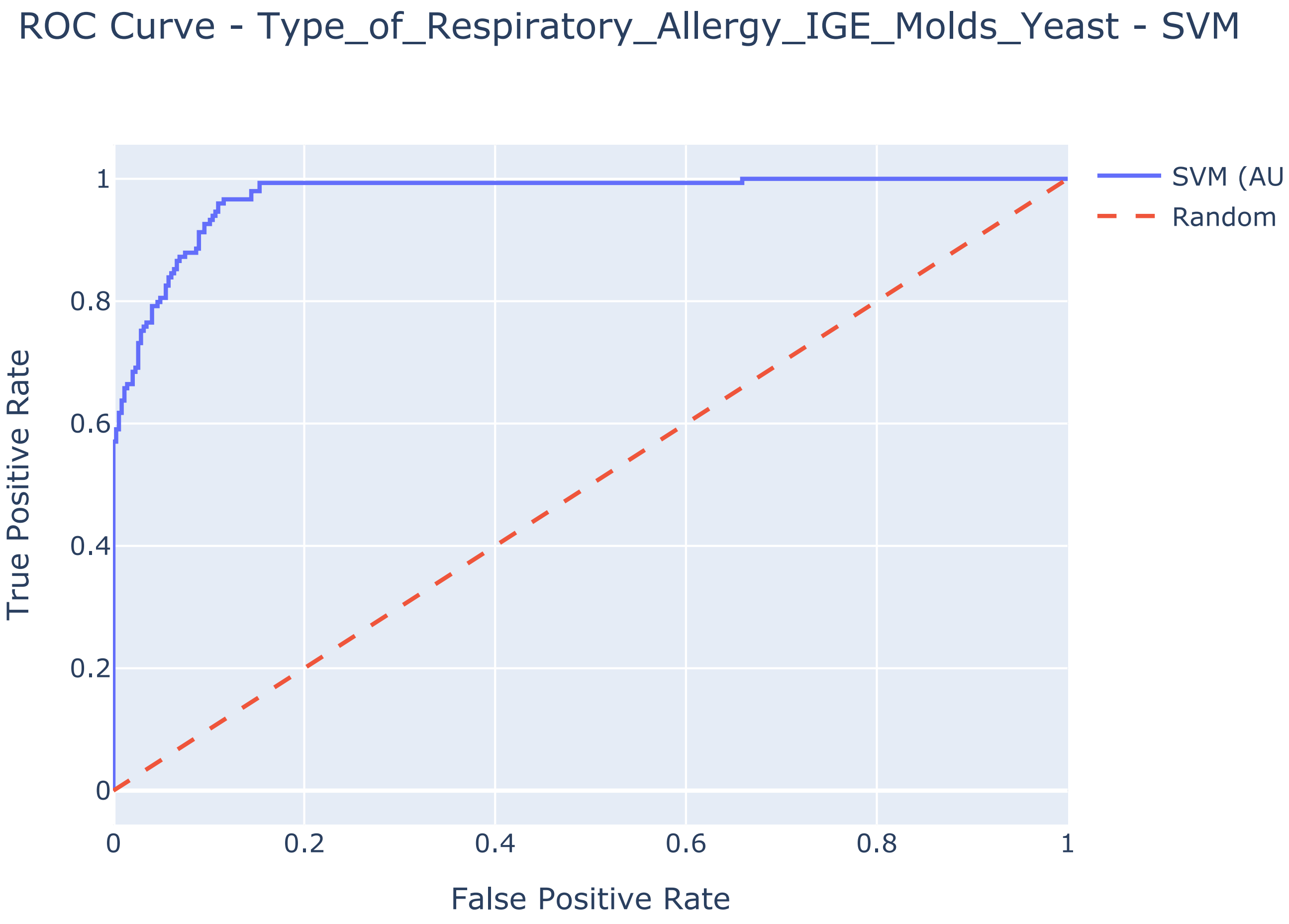


```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Molds_Yeast | Model: LogisticReg
📈 Accuracy: 0.8484
🎯 F1 (0): 0.8940 | F1 (1): 0.7301
📊 Precision: 0.8470 | AUC: 0.8622048819527812
📊 Confusion Matrix:
[[334  12]
 [ 40 109]]
```

ROC Curve - Type_of_Respiratory_Allergy_IGE_Molds_Yeast - Logisti

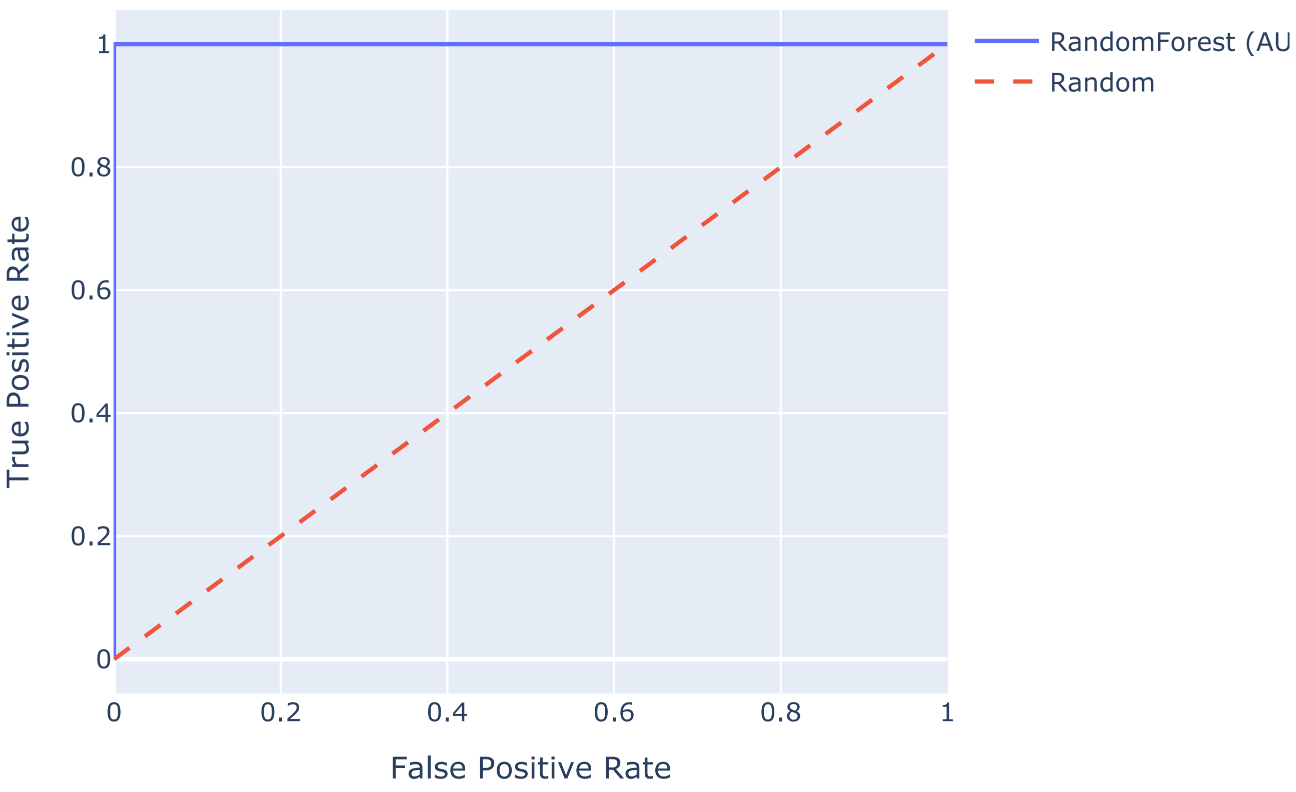


```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Molds_Yeast | Model: SVM
📈 Accuracy: 0.8405
🎯 F1 (0): 0.8927 | F1 (1): 0.6870
📊 Precision: 0.8403 | AUC: 0.8768483393357343
📊 Confusion Matrix:
 [[343   3]
 [ 57  92]]
```



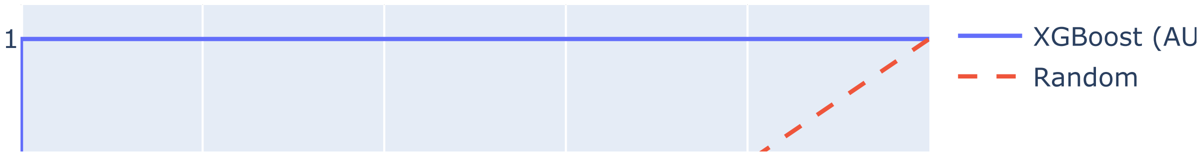
```
🔍 Target: Type_of_Respiratory_Allergy_ARIA | Model: RandomForest
🔍 Target: Type_of_Respiratory_Allergy_ARIA | Model: RandomForest
📈 Accuracy: 0.9737
🎯 F1 (0): 0.9646 | F1 (1): 0.9790
📊 Precision: 0.9757 | AUC: 0.9982993774759479
📊 Confusion Matrix:
 [[190   0]
 [  0 305]]
```

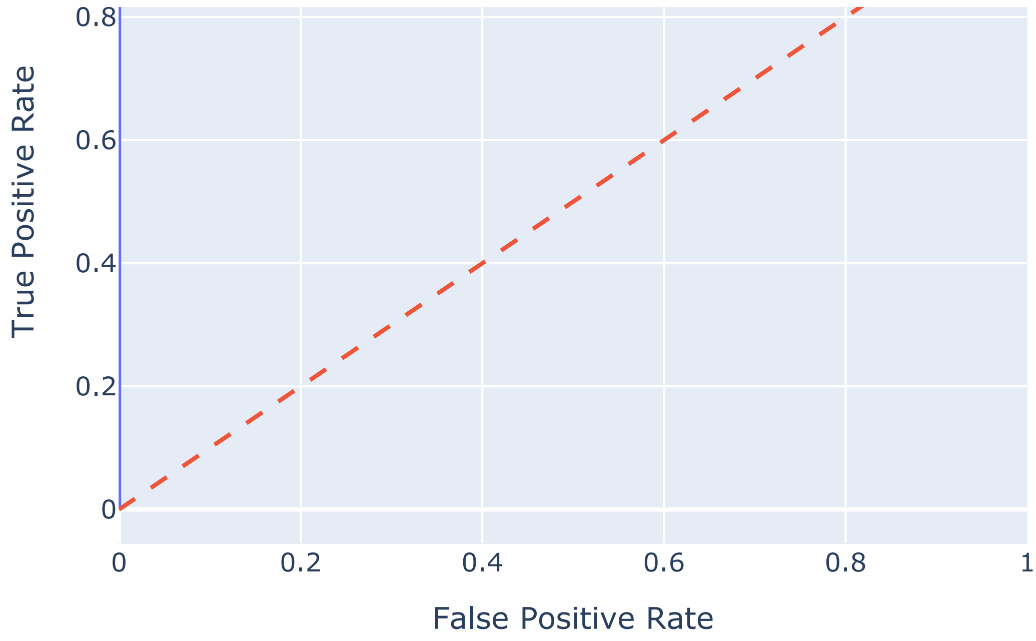
ROC Curve - Type_of_Respiratory_Allergy_ARIA - RandomForest



```
🔍 Target: Type_of_Respiratory_Allergy_ARIA | Model: XGBoost
📈 Accuracy: 1.0000
🎯 F1 (0): 1.0000 | F1 (1): 1.0000
📊 Precision: 1.0000 | AUC: 1.0
📊 Confusion Matrix:
 [[190   0]
 [  0 305]]
```

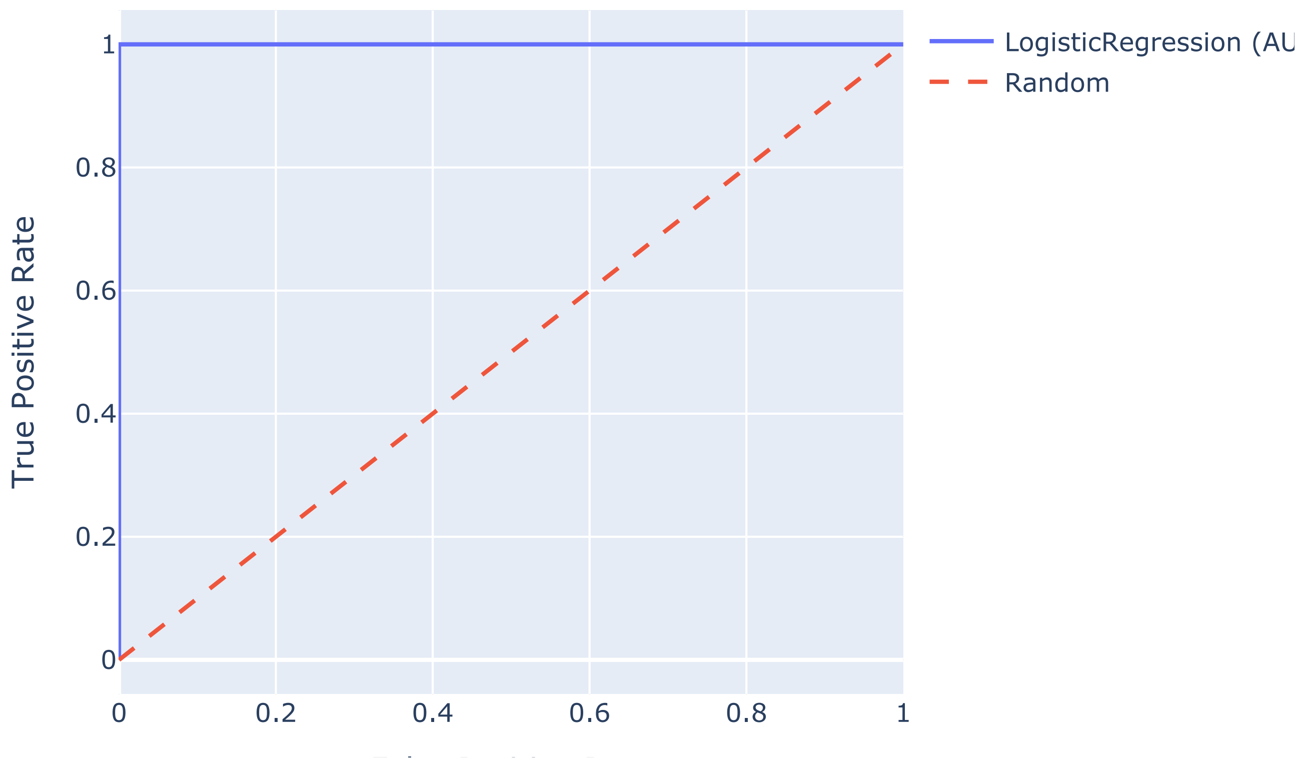
ROC Curve - Type_of_Respiratory_Allergy_ARIA - XGBoost





```
🔍 Target: Type_of_Respiratory_Allergy_ARIA | Model: LogisticRegression
📈 Accuracy: 0.9920
🎯 F1 (0): 0.9900 | F1 (1): 0.9933
📊 Precision: 0.9928 | AUC: 1.0
📊 Confusion Matrix:
 [[190   0]
 [  0 305]]
```

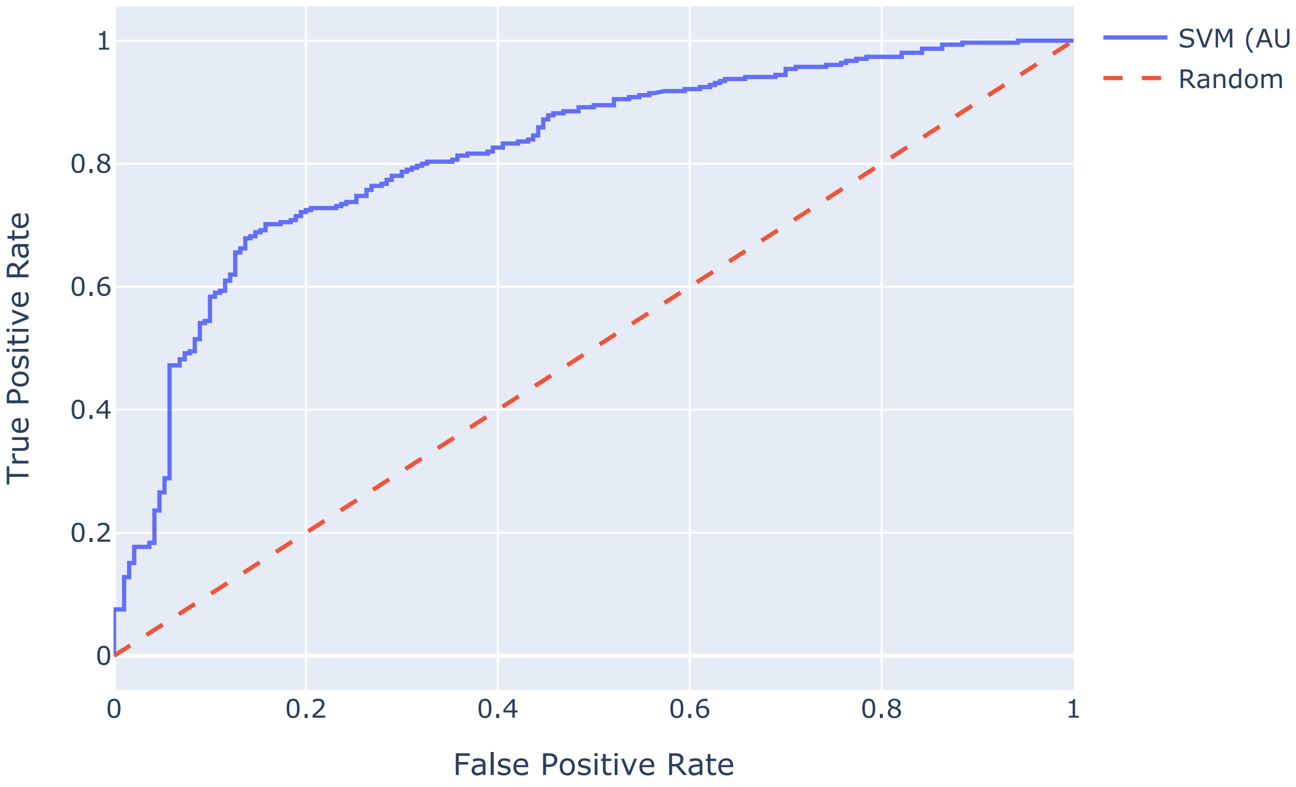
ROC Curve - Type_of_Respiratory_Allergy_ARIA - LogisticRegression



False Positive Rate

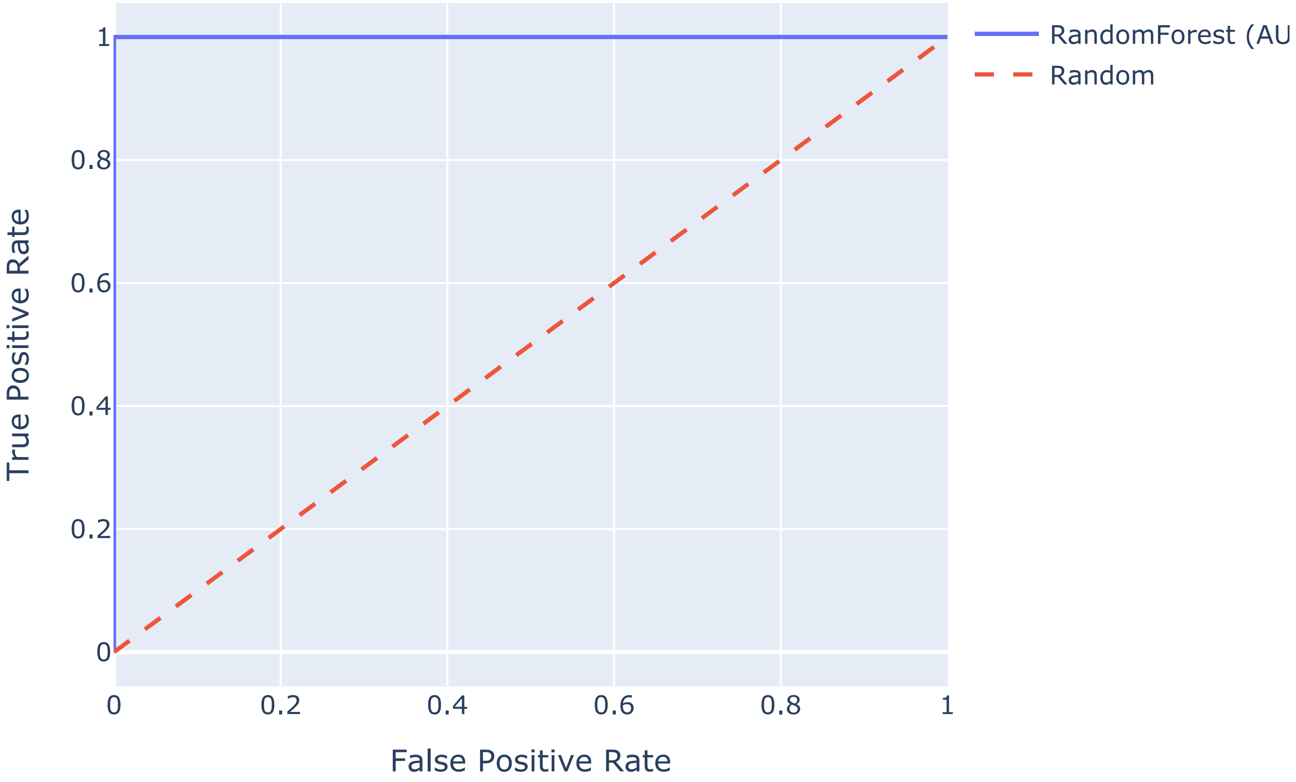
```
🔍 Target: Type_of_Respiratory_Allergy_ARIA | Model: SVM
📈 Accuracy: 0.6405
🎯 F1 (0): 0.5437 | F1 (1): 0.7016
📊 Precision: 0.6473 | AUC: 0.6669269949066214
📊 Confusion Matrix:
 [[ 72 118]
 [ 22 283]]
```

ROC Curve - Type_of_Respiratory_Allergy_ARIA - SVM



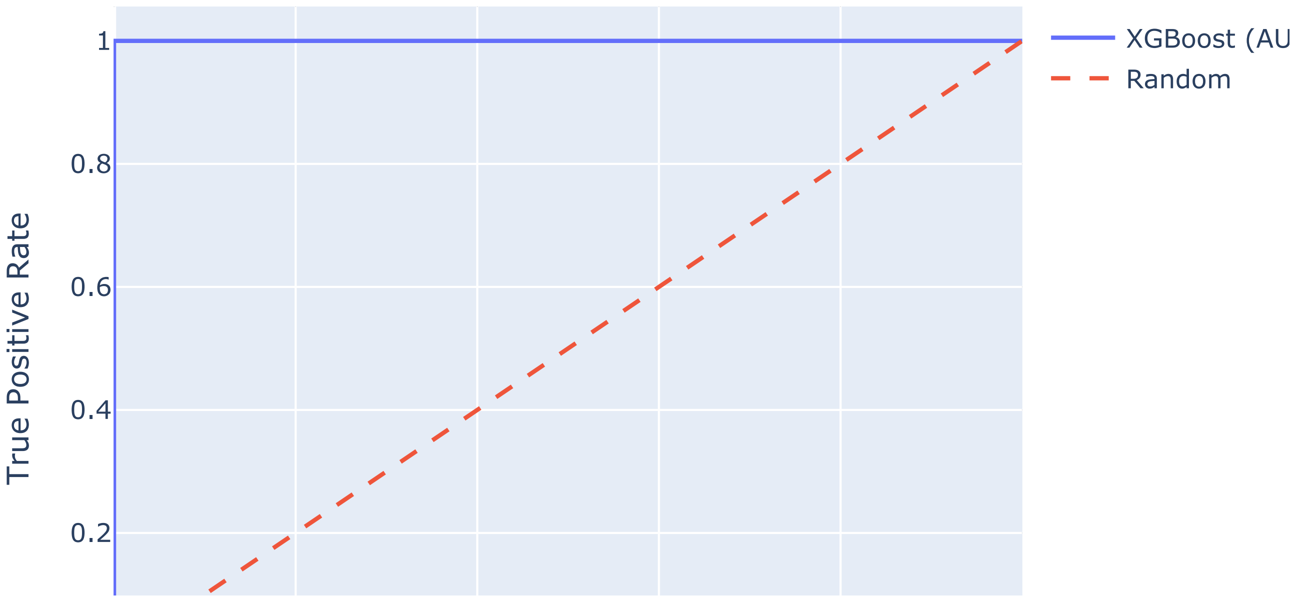
```
🔍 Target: Type_of_Respiratory_Allergy_CONJ | Model: RandomForest
📈 Accuracy: 0.9535
🎯 F1 (0): 0.9683 | F1 (1): 0.9122
📊 Precision: 0.9541 | AUC: 0.9940273417059131
📊 Confusion Matrix:
 [[356   0]
 [  0 139]]
```

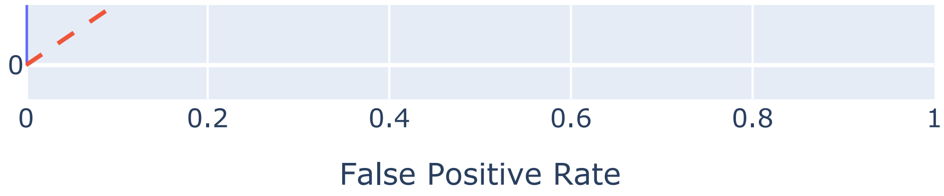
ROC Curve - Type_of_Respiratory_Allergy_CONJ - RandomForest



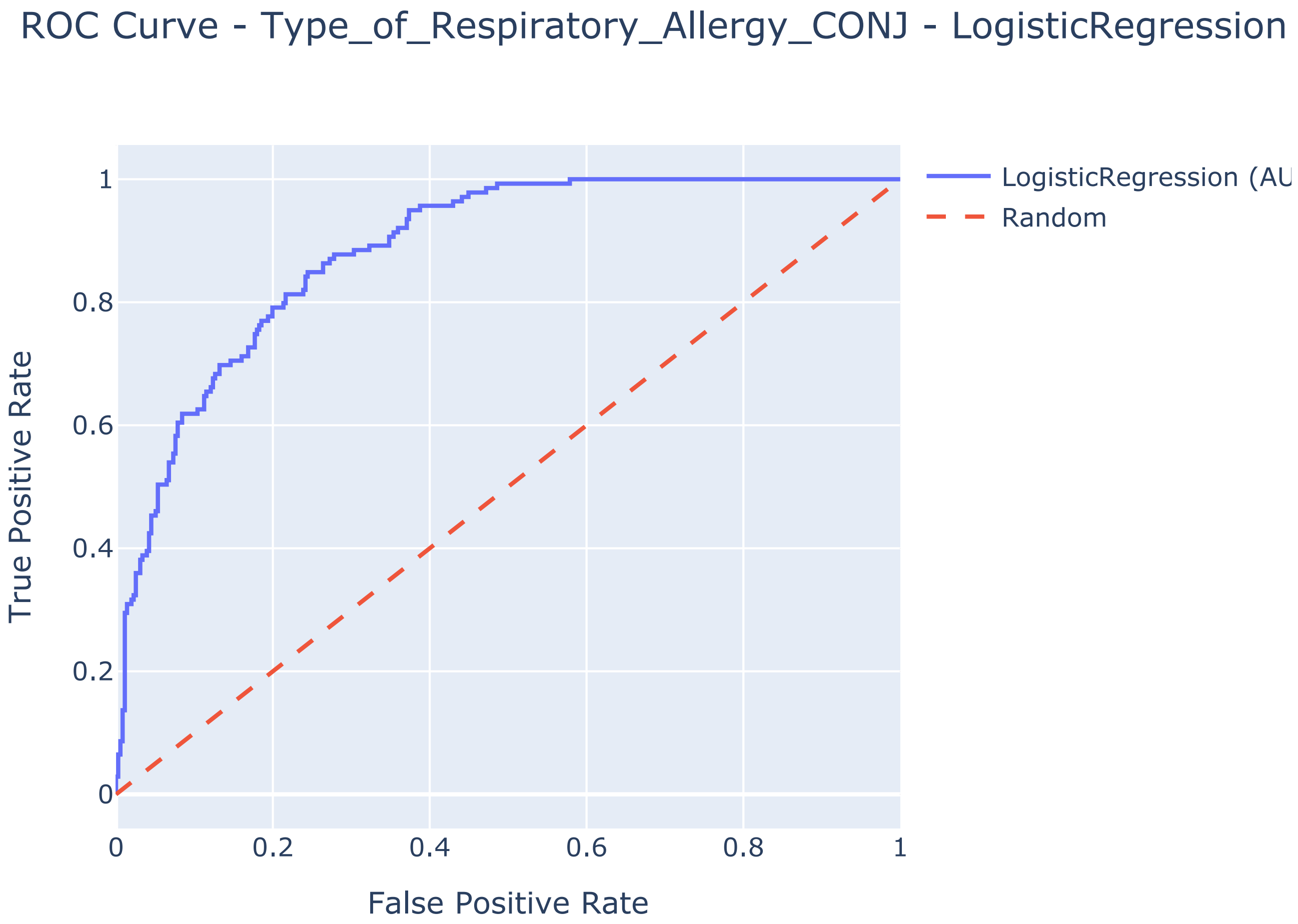
```
🔍 Target: Type_of_Respiratory_Allergy_CONJ | Model: XGBoost
📈 Accuracy: 1.0000
🎯 F1 (0): 1.0000 | F1 (1): 1.0000
📊 Precision: 1.0000 | AUC: 1.0
📊 Confusion Matrix:
 [[356   0]
 [  0 139]]
```

ROC Curve - Type_of_Respiratory_Allergy_CONJ - XGBoost

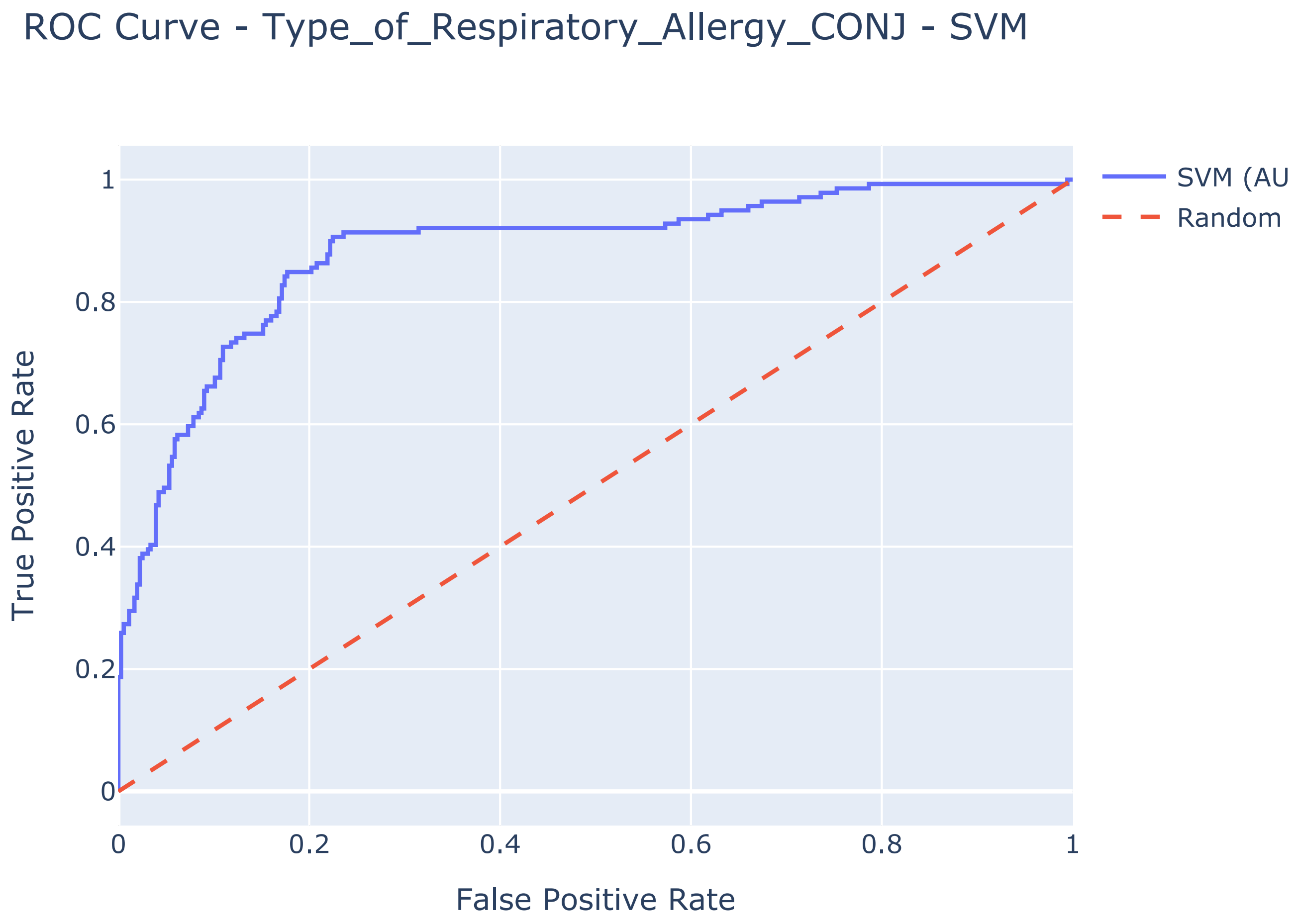




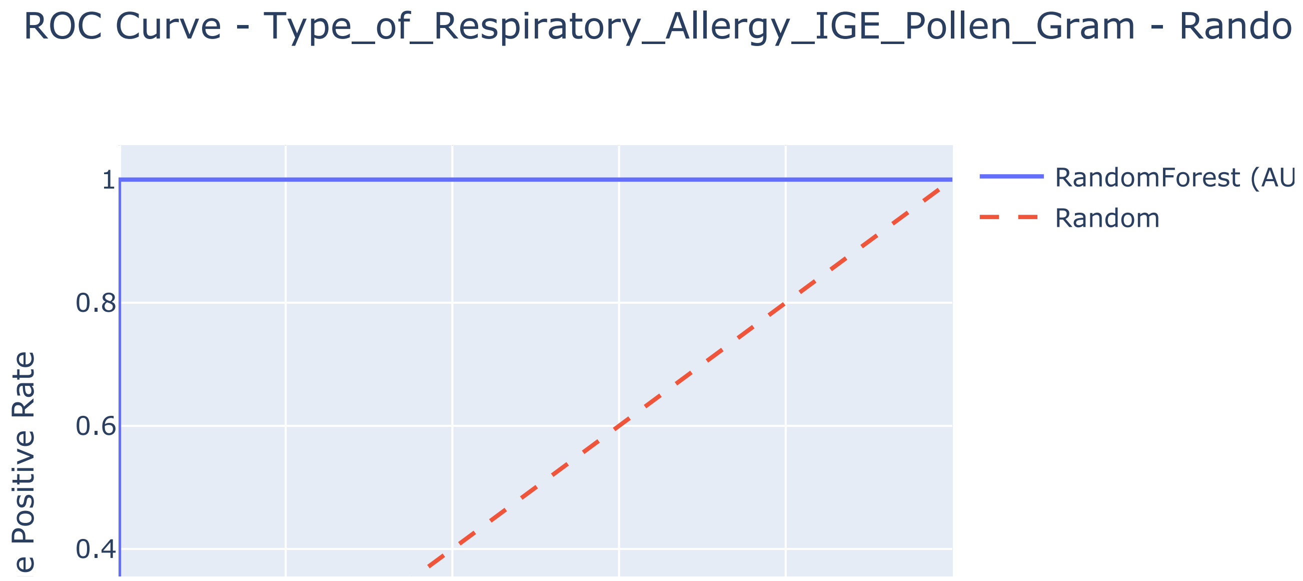
```
🔍 Target: Type_of_Respiratory_Allergy_CONJ | Model: LogisticRegression
📈 Accuracy: 0.7050
🎯 F1 (0): 0.7848 | F1 (1): 0.5221
📊 Precision: 0.7273 | AUC: 0.7483228676085819
📊 Confusion Matrix:
 [[329  27]
 [ 61  78]]
```



```
🔍 Target: Type_of_Respiratory_Allergy_CONJ | Model: SVM
📈 Accuracy: 0.6384
🎯 F1 (0): 0.7105 | F1 (1): 0.5080
📊 Precision: 0.7085 | AUC: 0.6739218123146694
📊 Confusion Matrix:
 [[356   0]
 [137   2]]
```

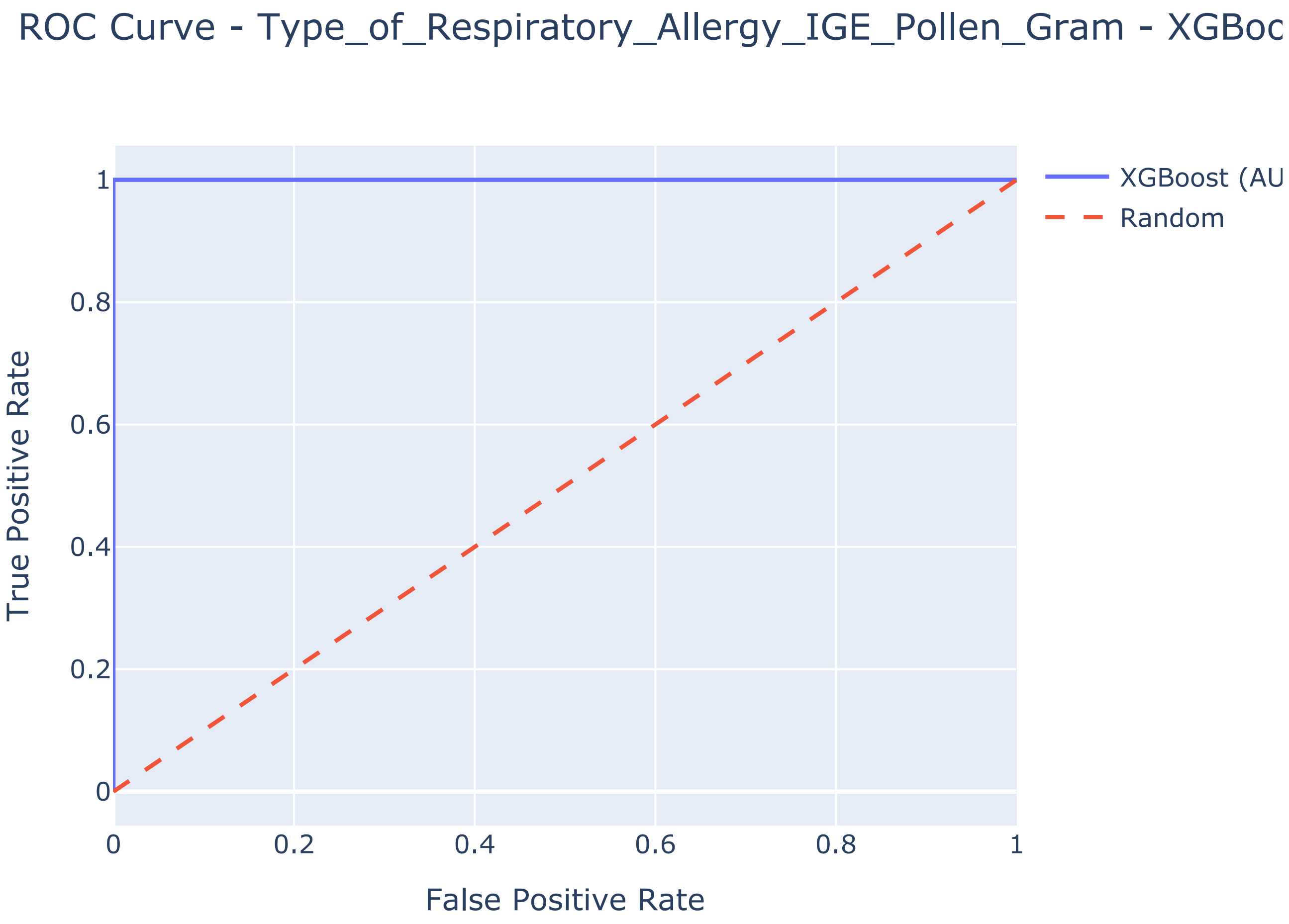


```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Pollen_Gram | Model: RandomFores
📈 Accuracy: 0.9436
🎯 F1 (0): 0.9214 | F1 (1): 0.9558
📊 Precision: 0.9468 | AUC: 0.974568006036597
📊 Confusion Matrix:
 [[187   0]
 [  0 308]]
```

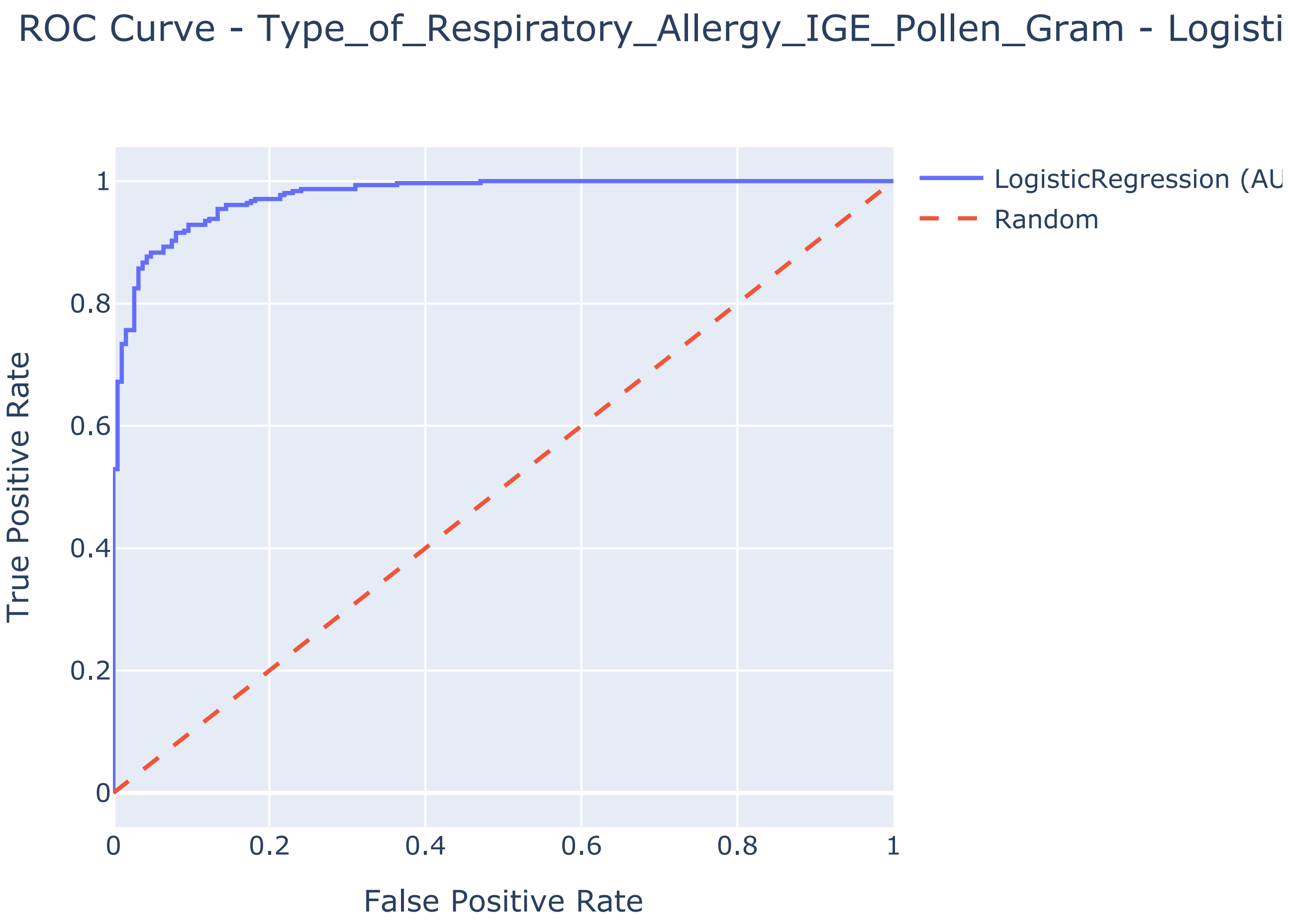


Tru
0.2
0
0
0.2
0.4
0.6
0.8
1
False Positive Rate

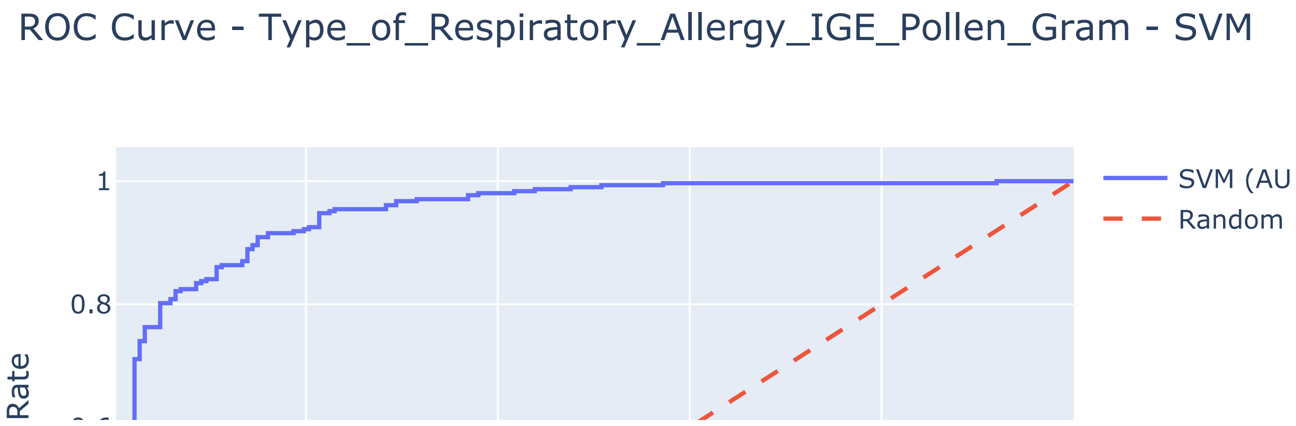
```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Pollen_Gram | Model: XGBoost
📈 Accuracy: 0.9274
🎯 F1 (0): 0.9009 | F1 (1): 0.9425
📊 Precision: 0.9308 | AUC: 0.9777872099603849
📊 Confusion Matrix:
 [[187   0]
 [  0 308]]
```

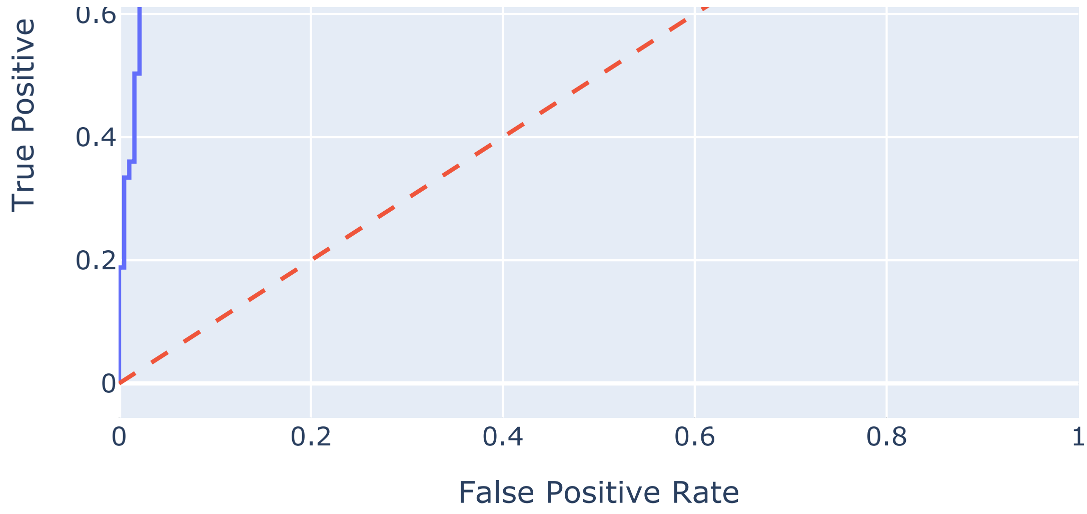


```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Pollen_Gram | Model: LogisticReg
📈 Accuracy: 0.8546
🎯 F1 (0): 0.8286 | F1 (1): 0.8730
📊 Precision: 0.8736 | AUC: 0.9187851348802113
📊 Confusion Matrix:
 [[173  14]
 [ 31 277]]
```

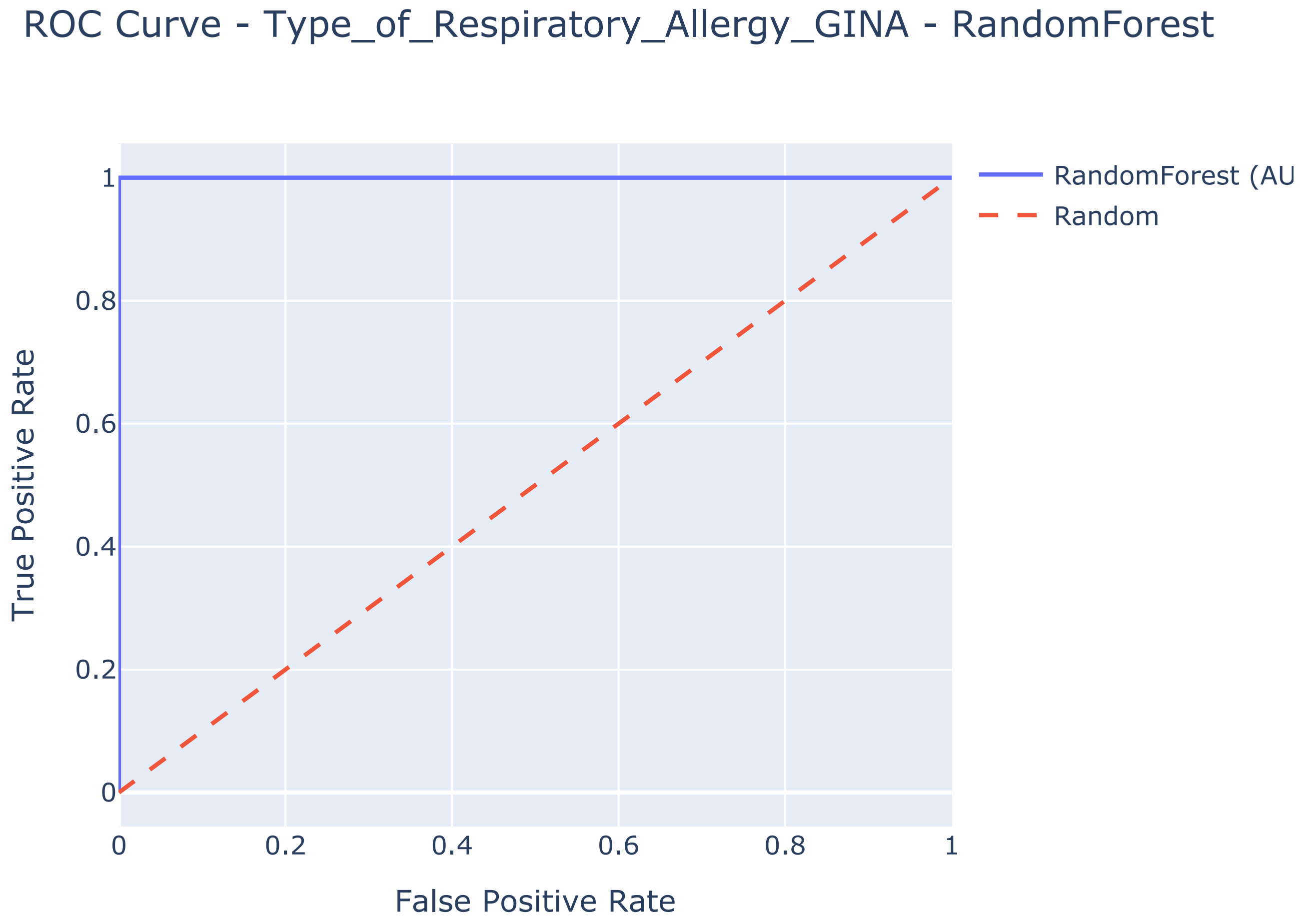

ROC Curve - Type_of_Respiratory_Allergy_IGE_Pollen_Gram - Logisti

```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Pollen_Gram | Model: SVM
📈 Accuracy: 0.8181
🎯 F1 (0): 0.7982 | F1 (1): 0.8336
📊 Precision: 0.8564 | AUC: 0.9313657800415015
📊 Confusion Matrix:
 [[162  25]
 [ 41 267]]
```

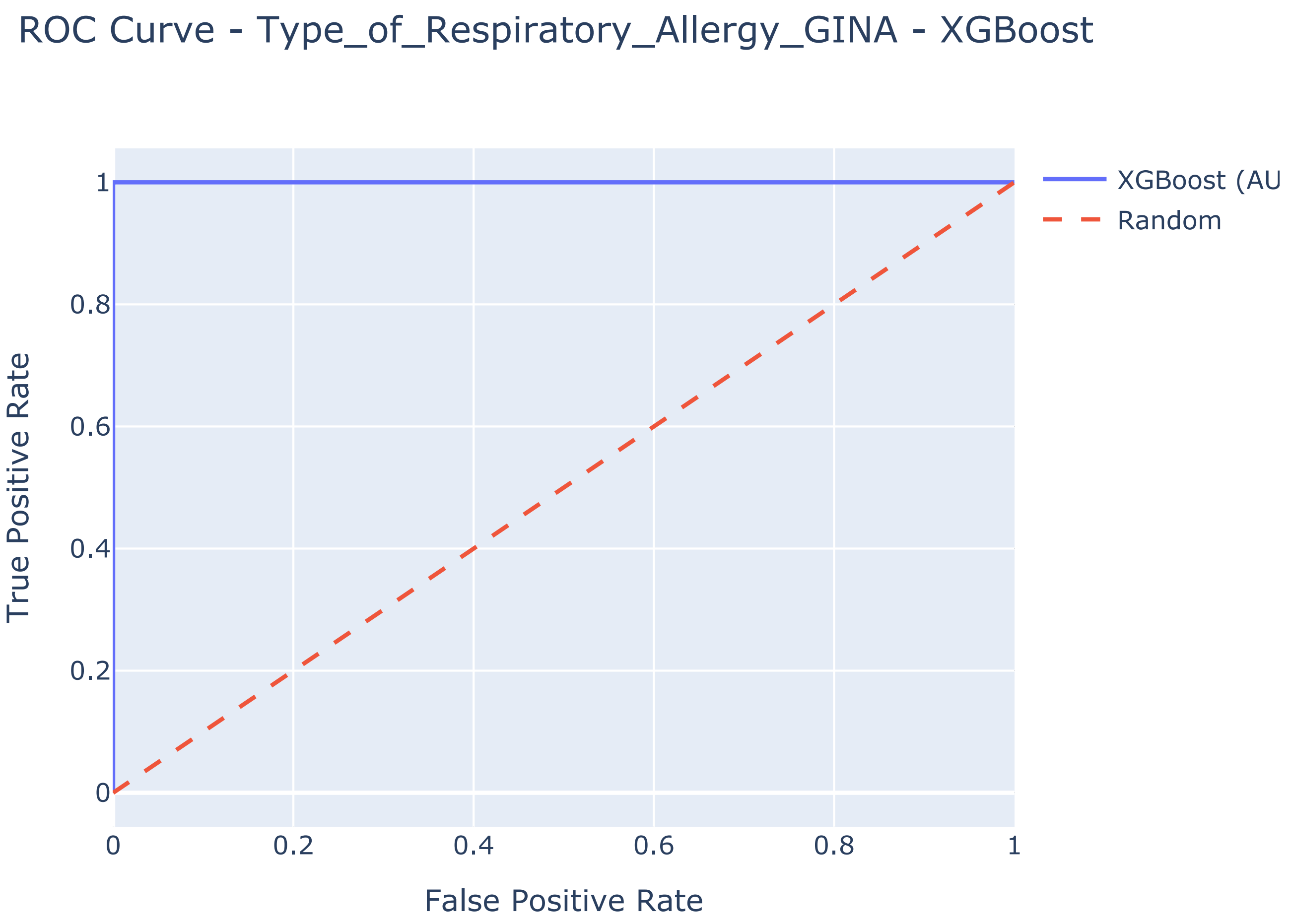

ROC Curve - Type_of_Respiratory_Allergy_IGE_Pollen_Gram - SVM



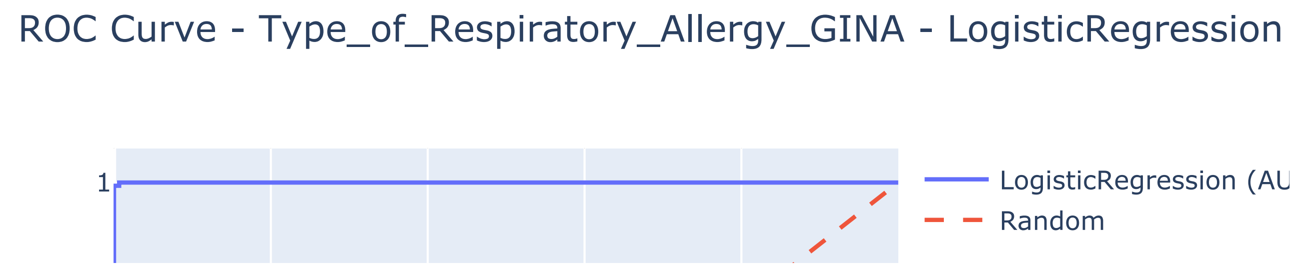
```
🔍 Target: Type_of_Respiratory_Allergy_GINA | Model: RandomForest
📈 Accuracy: 0.9130
🎯 F1 (0): 0.9261 | F1 (1): 0.8939
📊 Precision: 0.9204 | AUC: 0.9726219581211092
📊 Confusion Matrix:
 [[304   0]
 [  0 191]]
```

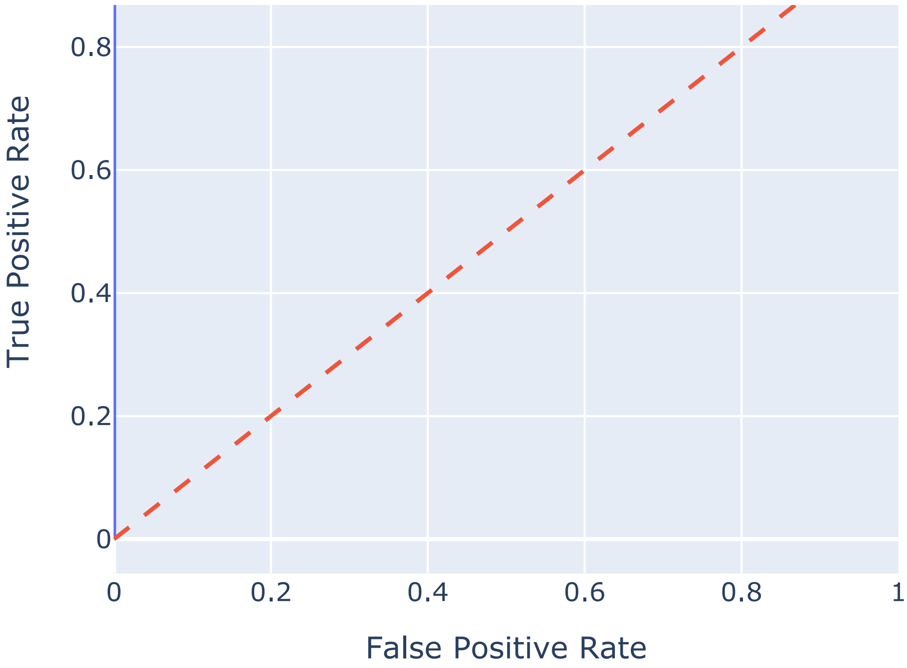


```
🔍 Target: Type_of_Respiratory_Allergy_GINA | Model: XGBoost
📈 Accuracy: 0.9231
🎯 F1 (0): 0.9347 | F1 (1): 0.9062
📊 Precision: 0.9289 | AUC: 0.9716921335597059
📊 Confusion Matrix:
 [[304   0]
 [  0 191]]
```

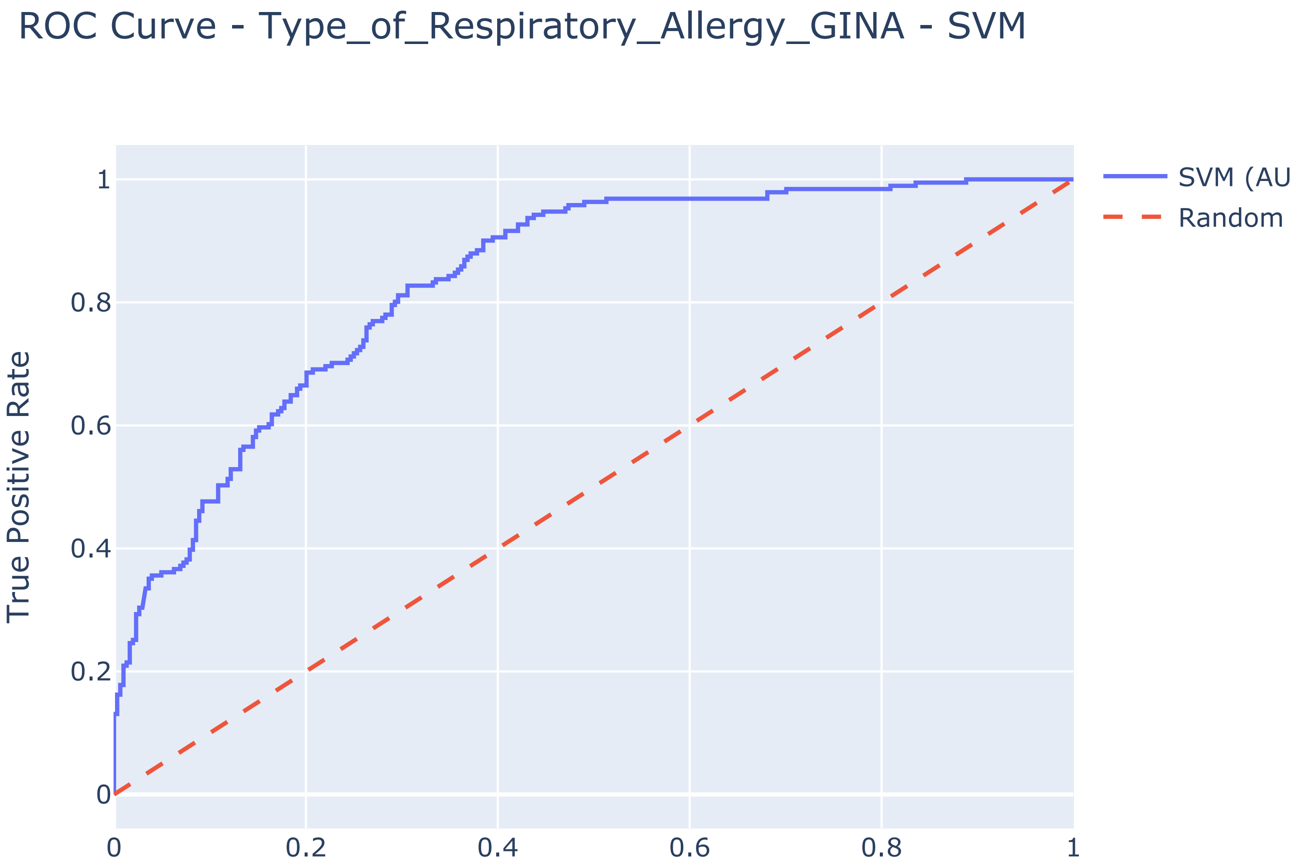

ROC Curve - Type_of_Respiratory_Allergy_GINA - XGBoost

```
🔍 Target: Type_of_Respiratory_Allergy_GINA | Model: LogisticRegression
📈 Accuracy: 0.9192
🎯 F1 (0): 0.9328 | F1 (1): 0.8983
📊 Precision: 0.9234 | AUC: 0.9801530843237126
📊 Confusion Matrix:
 [[304   0]
 [  2 189]]
```


ROC Curve - Type_of_Respiratory_Allergy_GINA - LogisticRegression



```
🔍 Target: Type_of_Respiratory_Allergy_GINA | Model: SVM
📈 Accuracy: 0.6202
🎯 F1 (0): 0.6776 | F1 (1): 0.5313
📊 Precision: 0.6309 | AUC: 0.6561621392190153
📊 Confusion Matrix:
 [[298   6]
 [143  48]]
```

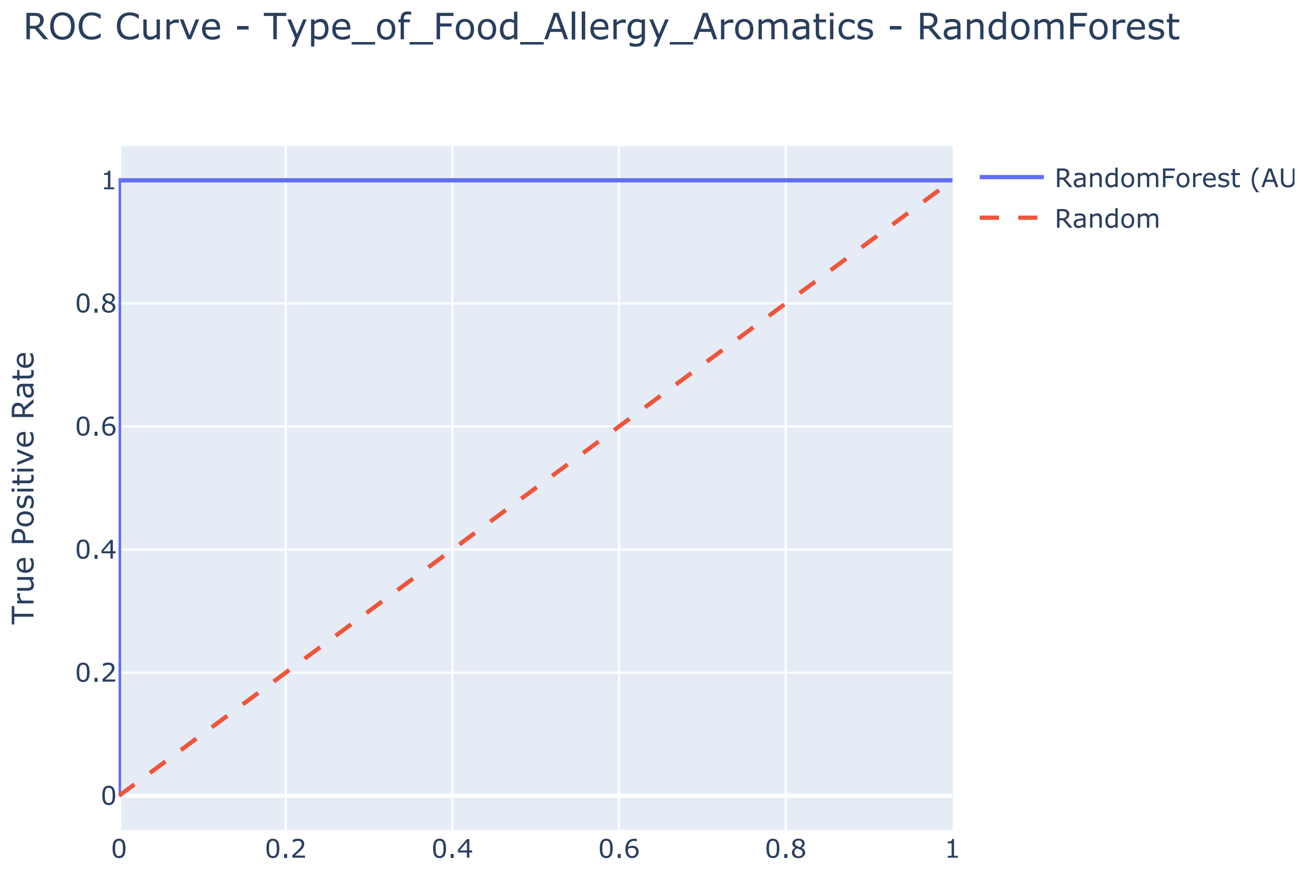


False Positive Rate

```
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedKFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.metrics import (
    f1_score, accuracy_score, recall_score,
    precision_score, confusion_matrix, roc_auc_score, roc_curve
)
from imblearn.over_sampling import SMOTE
import plotly.graph_objects as go

ALEX_food = ALEX[ALEX["Food_Allergy"] == 1]
targets = ["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"]

models = {
    "RandomForest": RandomForestClassifier(random_state=42),
    "XGBoost": XGBClassifier(random_state=42, eval_metric="logloss", use_label_
    "LogisticRegression": LogisticRegression(max_iter=1000, random_state=42),
    "SVM": SVC(probability=True, random_state=42)
}

X=ALEX_food.copy()
X.drop(target_1, axis=1, inplace=True)
X.drop(extra_columns, axis=1, inplace=True)
X.drop(extra, axis=1, inplace=True)
X.drop(inconnu, axis=1, inplace=True)
X = X.iloc[:, 1:]
results_food = []
```

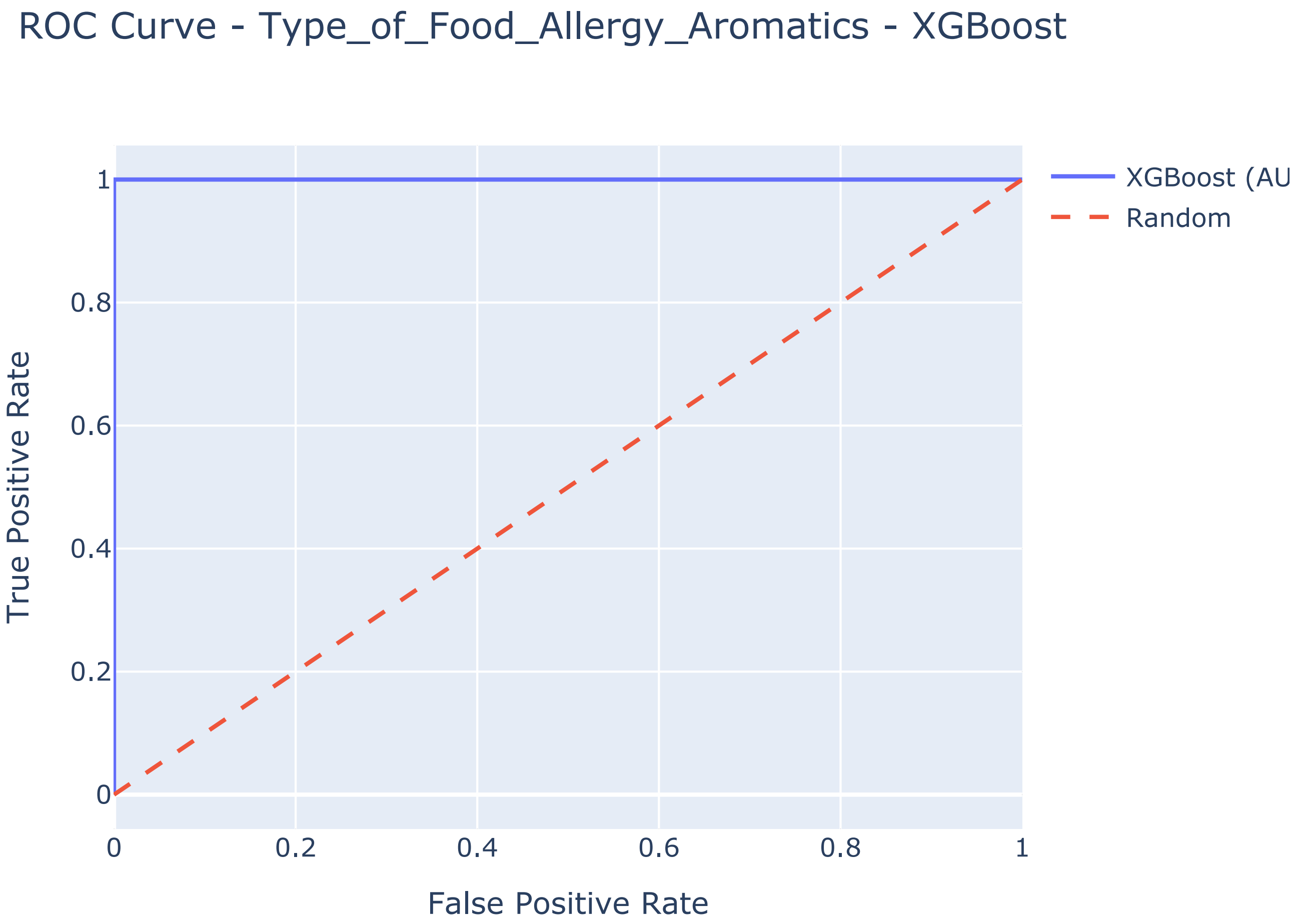
```
kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

for target in targets:
    y = ALEX_food[target]

    for model_name, base_model in models.items():
        f1_class0_scores, f1_class1_scores = [], []
        precision_scores, acc_scores, recall_scores, auc_scores = [], [], [], [

        for train_idx, test_idx in kfold.split(X, y):
            X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
            y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]

            smote = SMOTE(random_state=42)
            X_train_res, y_train_res = smote.fit_resample(X_train, y_train)

            base_model.fit(X_train_res, y_train_res)
            y_pred = base_model.predict(X_test)

            acc_scores.append(accuracy_score(y_test, y_pred))
            recall_scores.append(recall_score(y_test, y_pred, zero_division=0))
            precision_scores.append(precision_score(y_test, y_pred, average='we
            f1_class0_scores.append(f1_score(y_test, y_pred, pos_label=0, zero_
            f1_class1_scores.append(f1_score(y_test, y_pred, pos_label=1, zero_

            if hasattr(base_model, "predict_proba"):
                y_proba = base_model.predict_proba(X_test)[:, 1]
                auc_scores.append(roc_auc_score(y_test, y_proba))

        base_model.fit(X, y)
        y_pred_full = base_model.predict(X)
        y_proba_full = base_model.predict_proba(X)[:, 1] if hasattr(base_model,
        matrix = confusion_matrix(y, y_pred_full)

        print(f"\n🔍 Target: {target} | Model: {model_name}")
        print(f"📈 Accuracy: {np.mean(acc_scores):.4f}")
        print(f"🎯 F1 (0): {np.mean(f1_class0_scores):.4f} | F1 (1): {np.mean(
        print(f"📊 Precision: {np.mean(precision_scores):.4f} | AUC: {np.mean(
        print("📊 Confusion Matrix:\n", matrix)

        if y_proba_full is not None:
            fpr, tpr, _ = roc_curve(y, y_proba_full)
            fig = go.Figure()
            fig.add_trace(go.Scatter(x=fpr, y=tpr, mode='lines', name=f"{model_
            fig.add_trace(go.Scatter(x=[0, 1], y=[0, 1], mode='lines', name='Ra
            fig.update_layout(
                title=f"ROC Curve - {target} - {model_name}",
                xaxis_title="False Positive Rate",
                yaxis_title="True Positive Rate",
                width=700, height=500
            )
            fig.show()

        results_food.append({
            "Target": target,
            "Model": model_name,
            "F1_Class_0": np.mean(f1_class0_scores),
            "F1_Class_1": np.mean(f1_class1_scores),
            "Precision": np.mean(precision_scores),
            "Accuracy": np.mean(acc_scores),
            "Recall": np.mean(recall_scores),
            "AUC_ROC": np.mean(auc_scores) if auc_scores else np.nan
        })

pd.DataFrame(results_food).to_csv("results_ALEX_food.csv", index=False)
```

```
🔍 Target: Type_of_Food_Allergy_Aromatics | Model: RandomForest
📈 Accuracy: 0.9251
🎯 F1 (0): 0.9608 | F1 (1): 0.0400
📊 Precision: 0.8756 | AUC: 0.744212962962963
📊 Confusion Matrix:
 [[360   0]
 [  0  26]]
```



False Positive Rate

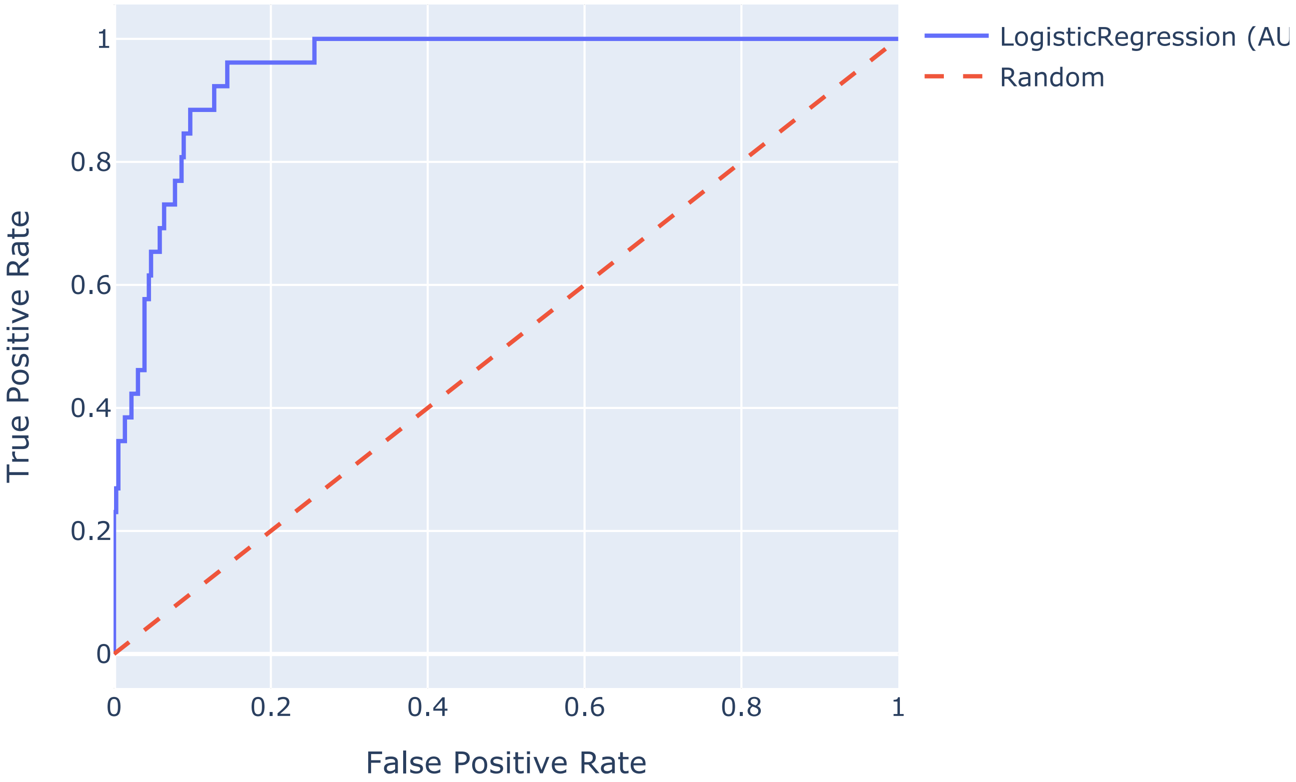
```
🔍 Target: Type_of_Food_Allergy_Aromatics | Model: XGBoost
📈 Accuracy: 0.9017
🎯 F1 (0): 0.9474 | F1 (1): 0.1733
📊 Precision: 0.8936 | AUC: 0.7592592592592593
📊 Confusion Matrix:
 [[360   0]
 [  0  26]]
```



ROC Curve - Type_of_Food_Allergy_Aromatics - XGBoost

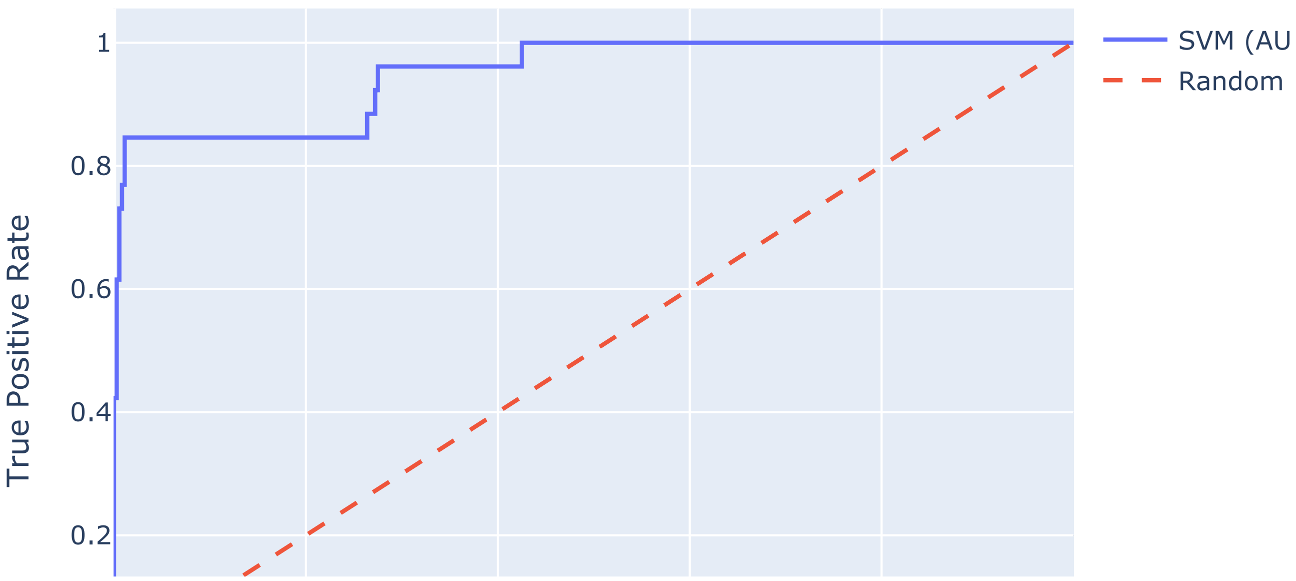
```
🔍 Target: Type_of_Food_Allergy_Aromatics | Model: LogisticRegression
📈 Accuracy: 0.8707
🎯 F1 (0): 0.9296 | F1 (1): 0.1436
📊 Precision: 0.8857 | AUC: 0.7171296296296296
📊 Confusion Matrix:
 [[359   1]
 [ 20   6]]
```

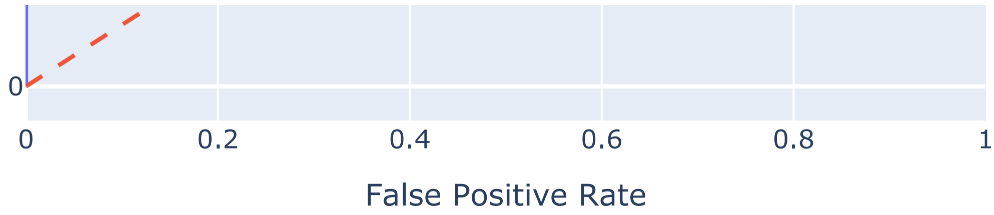
ROC Curve - Type_of_Food_Allergy_Aromatics - LogisticRegression



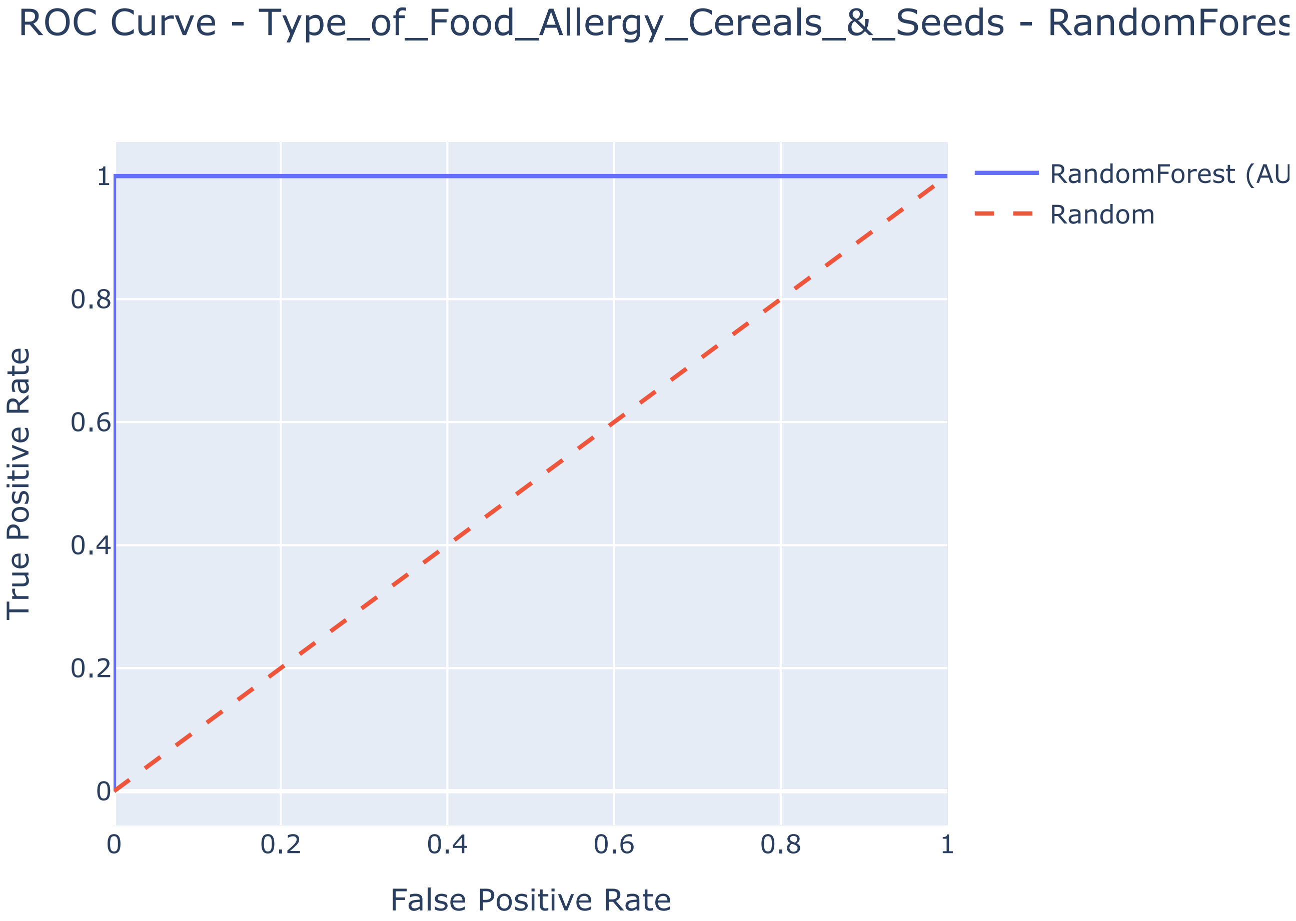
```
🔍 Target: Type_of_Food_Allergy_Aromatics | Model: SVM
📈 Accuracy: 0.7123
🎯 F1 (0): 0.8261 | F1 (1): 0.1400
📊 Precision: 0.8872 | AUC: 0.6351851851851852
📊 Confusion Matrix:
 [[360   0]
 [ 26   0]]
```

ROC Curve - Type_of_Food_Allergy_Aromatics - SVM

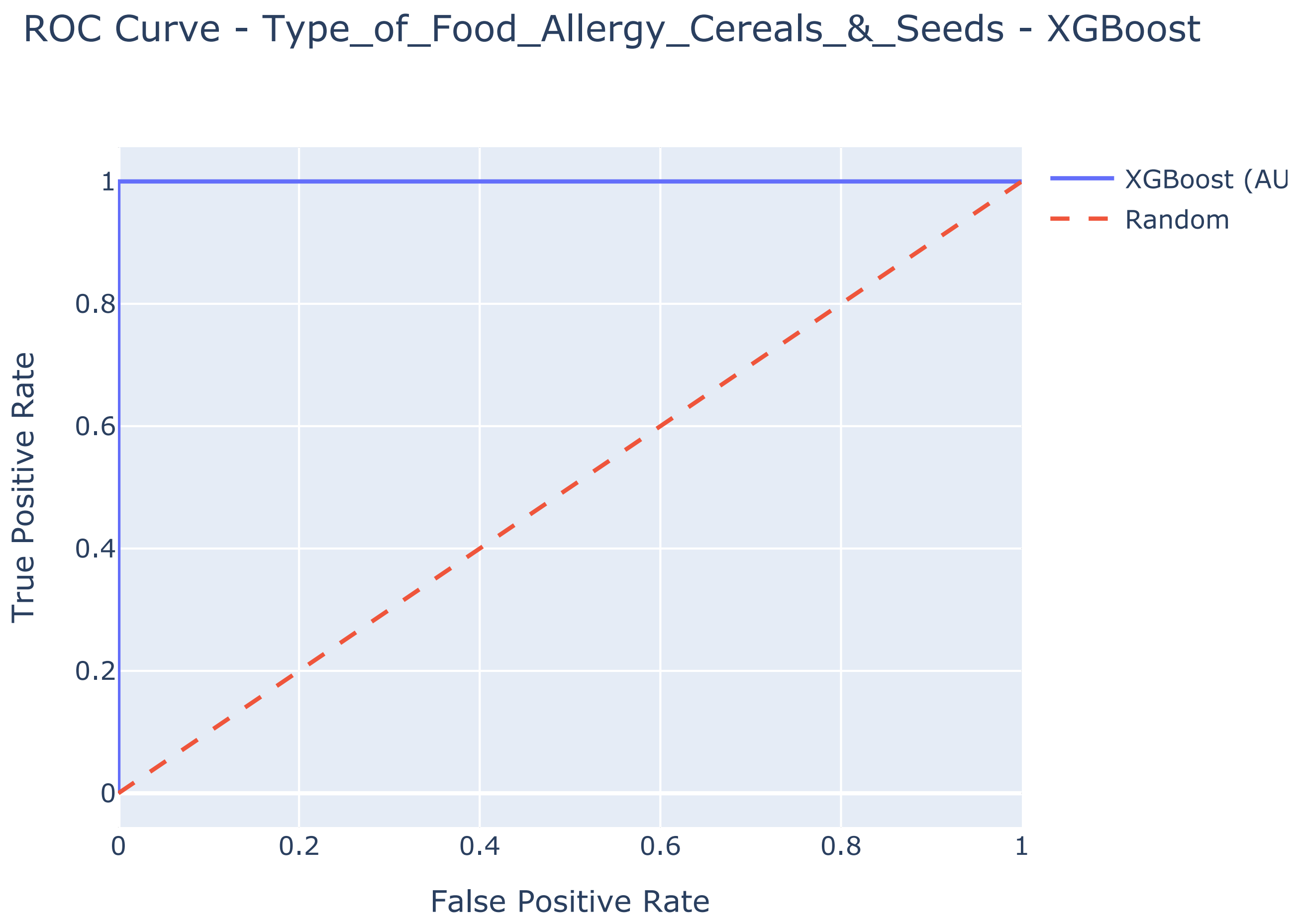




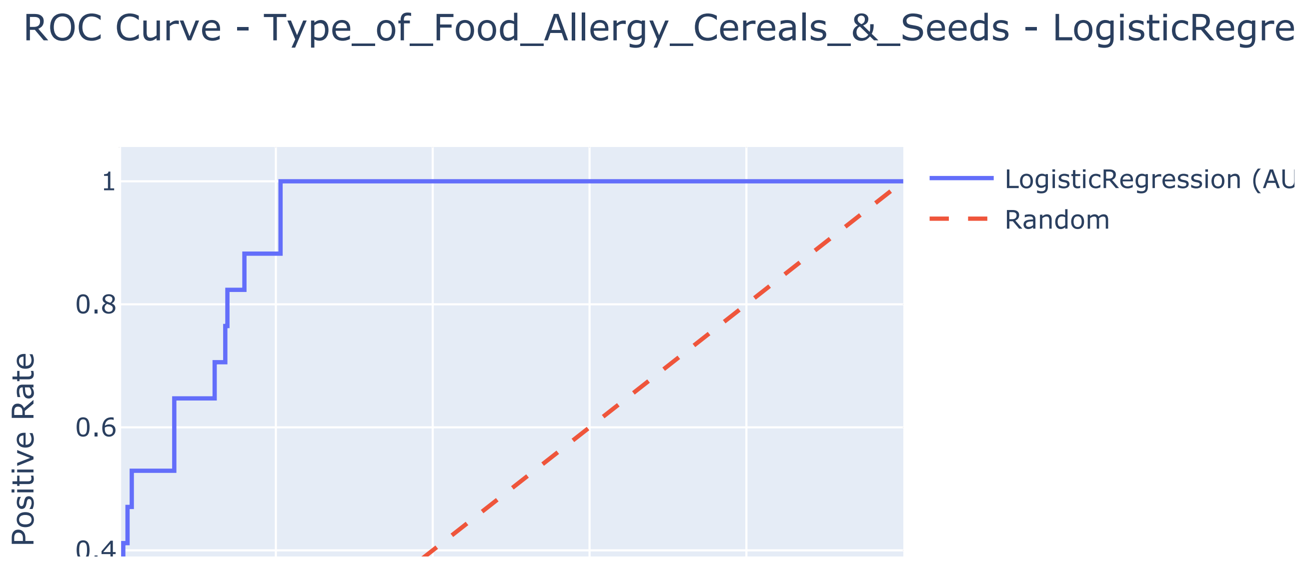
```
🔍 Target: Type_of_Food_Allergy_Cereals_&_Seeds | Model: RandomForest
📈 Accuracy: 0.9431
🎯 F1 (0): 0.9707 | F1 (1): 0.0000
📊 Precision: 0.9137 | AUC: 0.43384009009009006
📊 Confusion Matrix:
 [[369   0]
 [  0  17]]
```



```
🔍 Target: Type_of_Food_Allergy_Cereals_&_Seeds | Model: XGBoost
📈 Accuracy: 0.9096
🎯 F1 (0): 0.9523 | F1 (1): 0.0000
📊 Precision: 0.9120 | AUC: 0.4016516516516517
📊 Confusion Matrix:
 [[369   0]
 [  0  17]]
```

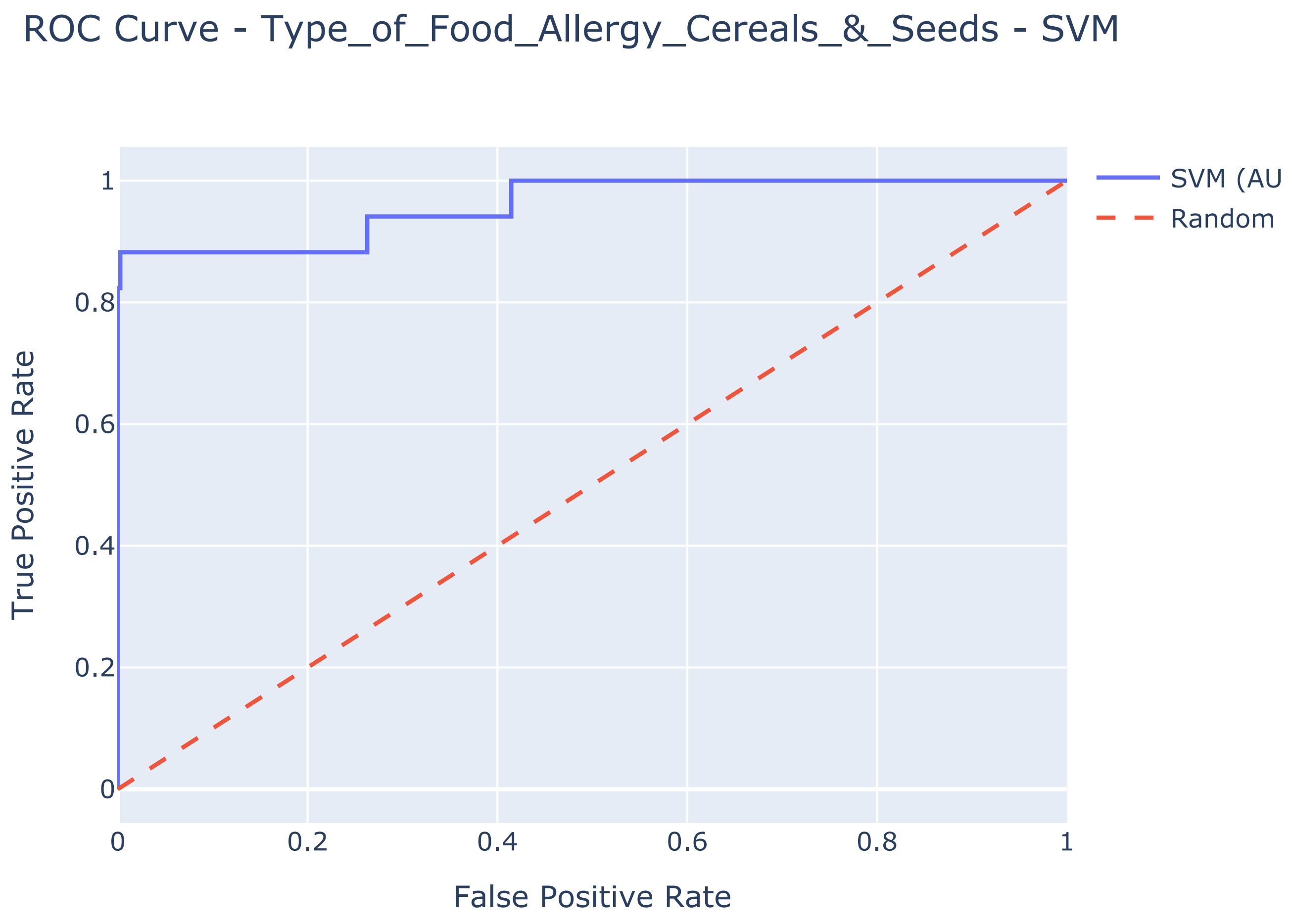


```
🔍 Target: Type_of_Food_Allergy_Cereals_&_Seeds | Model: LogisticRegression
📈 Accuracy: 0.8806
🎯 F1 (0): 0.9355 | F1 (1): 0.0867
📊 Precision: 0.9219 | AUC: 0.4336336336336336
📊 Confusion Matrix:
 [[369   0]
 [ 14   3]]
```



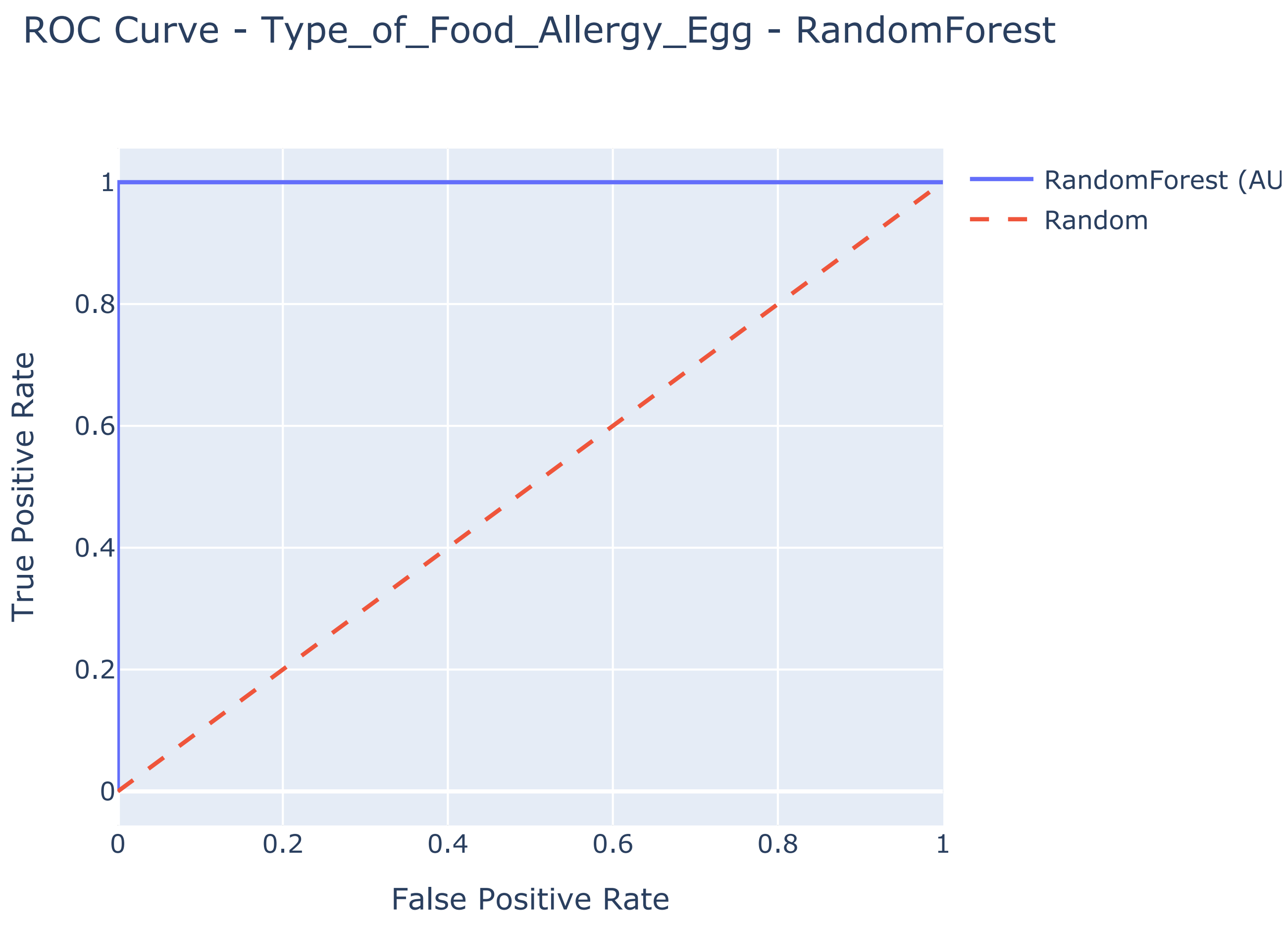
True
0.2
0
0
0.2
0.4
0.6
0.8
1
False Positive Rate

```
🔍 Target: Type_of_Food_Allergy_Cereals_&_Seeds | Model: SVM
📈 Accuracy: 0.8265
🎯 F1 (0): 0.9038 | F1 (1): 0.0686
📊 Precision: 0.9148 | AUC: 0.3957582582582583
📊 Confusion Matrix:
 [[369   0]
 [ 17   0]]
```

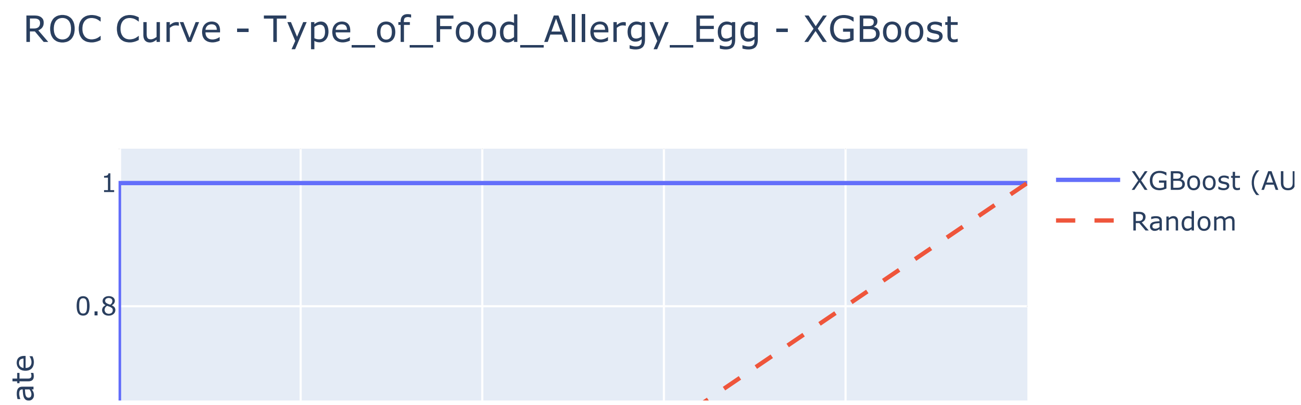


```
🔍 Target: Type_of_Food_Allergy_Egg | Model: RandomForest
📈 Accuracy: 0.8937
🎯 F1 (0): 0.9428 | F1 (1): 0.1500
```

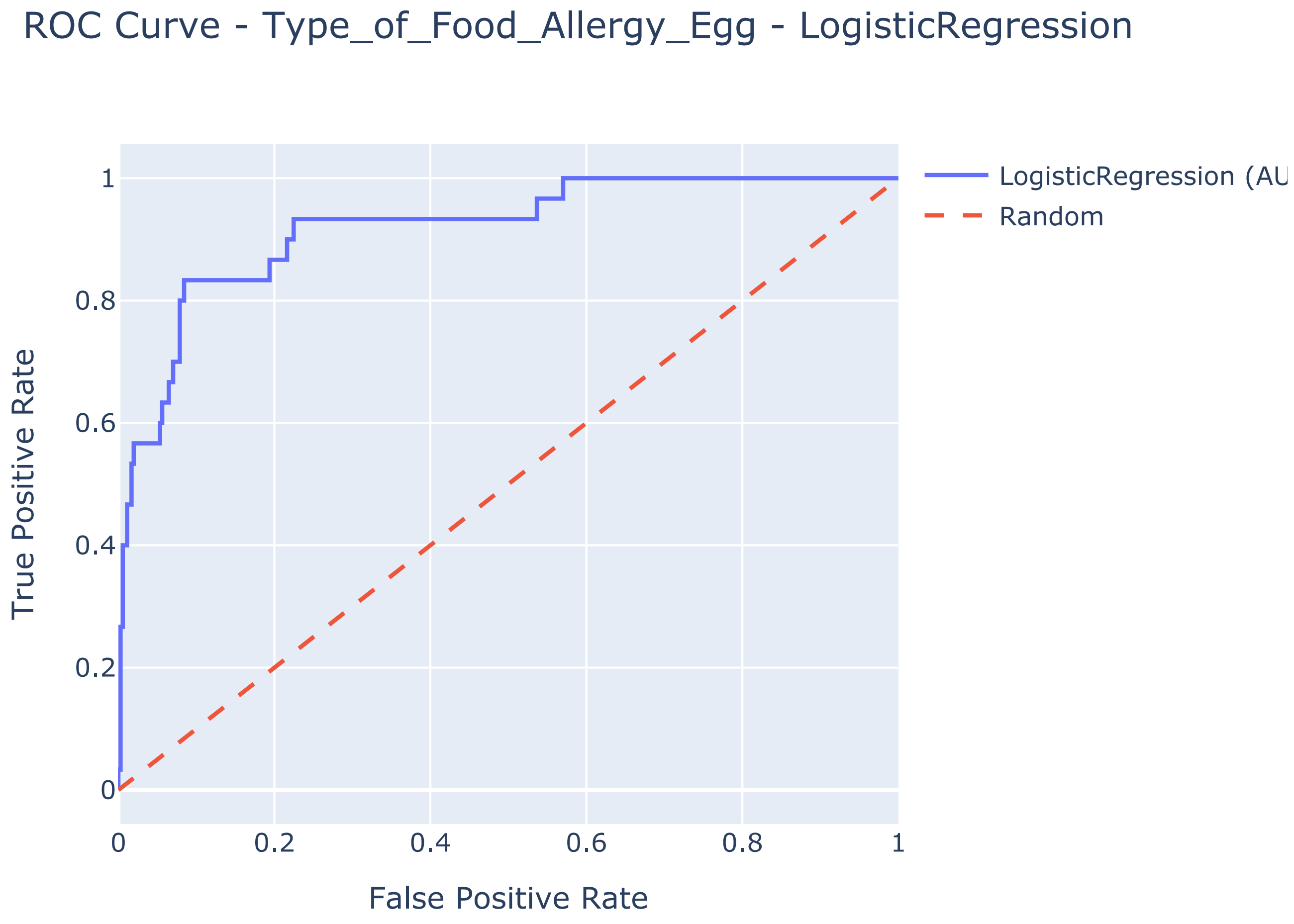
```
📊 Precision: 0.8756 | AUC: 0.6691269841269841
📊 Confusion Matrix:
 [[356   0]
 [  0  30]]
```



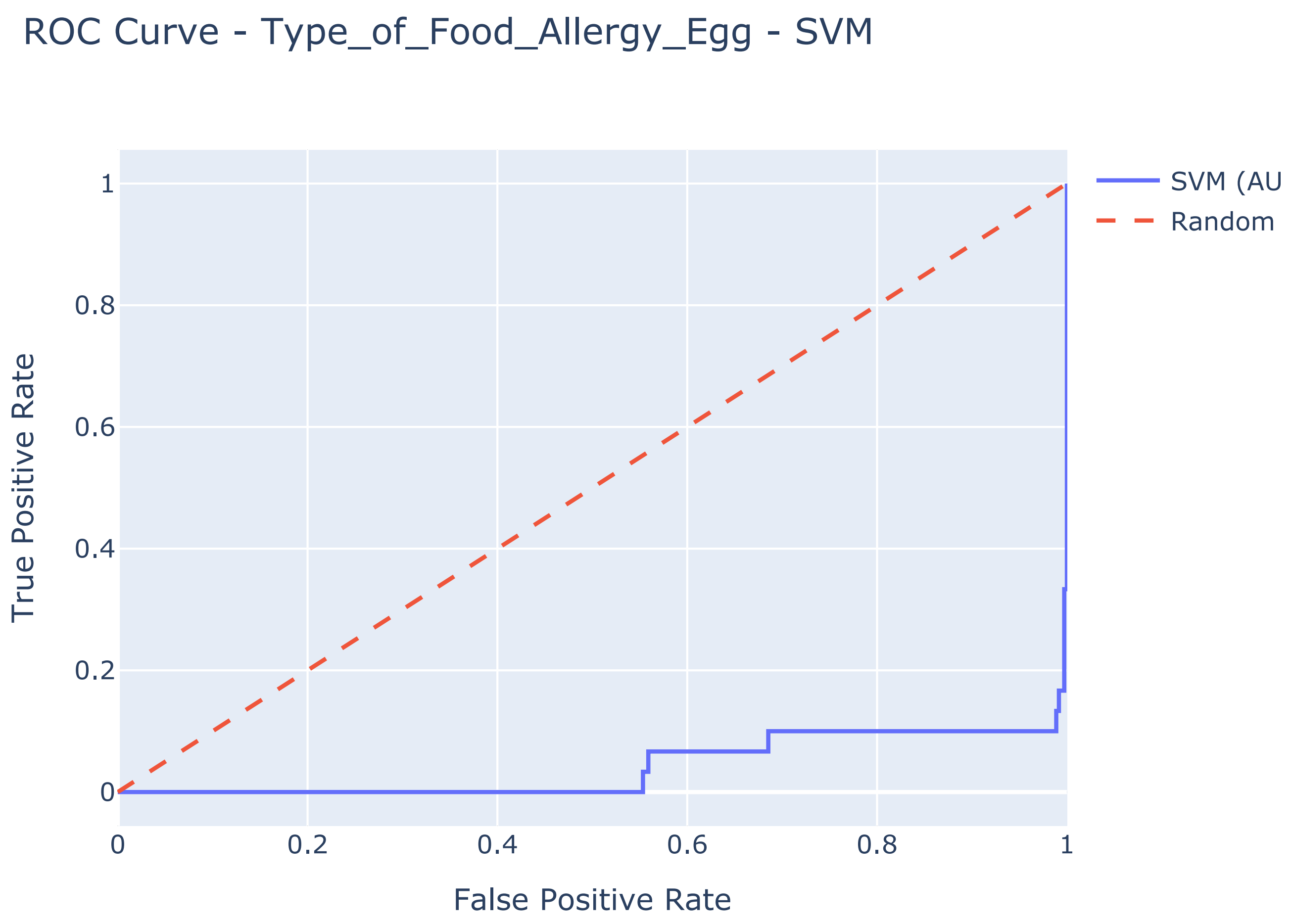
```
🔍 Target: Type_of_Food_Allergy_Egg | Model: XGBoost
📈 Accuracy: 0.8835
🎯 F1 (0): 0.9371 | F1 (1): 0.1550
📊 Precision: 0.8712 | AUC: 0.6938359788359788
📊 Confusion Matrix:
 [[356   0]
 [  0  30]]
```



```
🔍 Target: Type_of_Food_Allergy_Egg | Model: LogisticRegression
📈 Accuracy: 0.8501
🎯 F1 (0): 0.9167 | F1 (1): 0.2260
📊 Precision: 0.8796 | AUC: 0.6274603174603174
📊 Confusion Matrix:
 [[354   2]
 [ 21   9]]
```

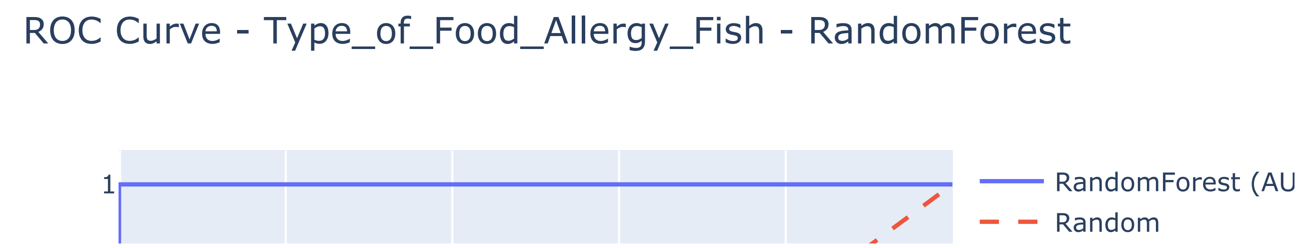

ROC Curve - Type_of_Food_Allergy_Egg - LogisticRegression

```
🔍 Target: Type_of_Food_Allergy_Egg | Model: SVM
📈 Accuracy: 0.8367
🎯 F1 (0): 0.9078 | F1 (1): 0.1990
📊 Precision: 0.8789 | AUC: 0.5953968253968254
📊 Confusion Matrix:
 [[356   0]
 [ 30   0]]
```

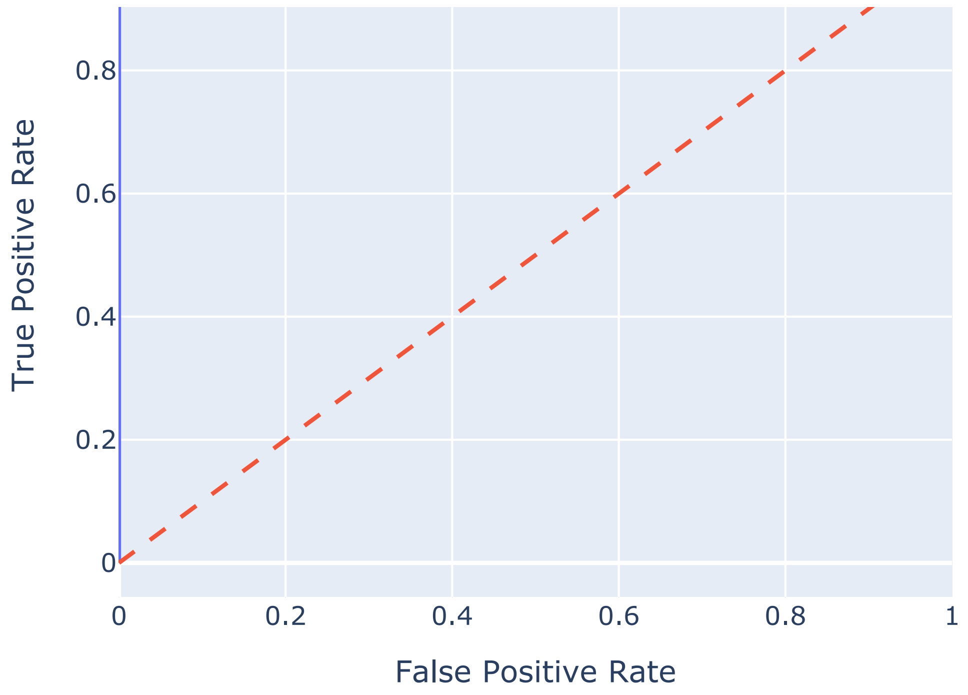


ROC Curve - Type_of_Food_Allergy_Egg - SVM

```
🔍 Target: Type_of_Food_Allergy_Fish | Model: RandomForest
📈 Accuracy: 0.9301
🎯 F1 (0): 0.9628 | F1 (1): 0.3871
📊 Precision: 0.9154 | AUC: 0.7351719576719578
📊 Confusion Matrix:
 [[356   0]
 [  0  30]]
```

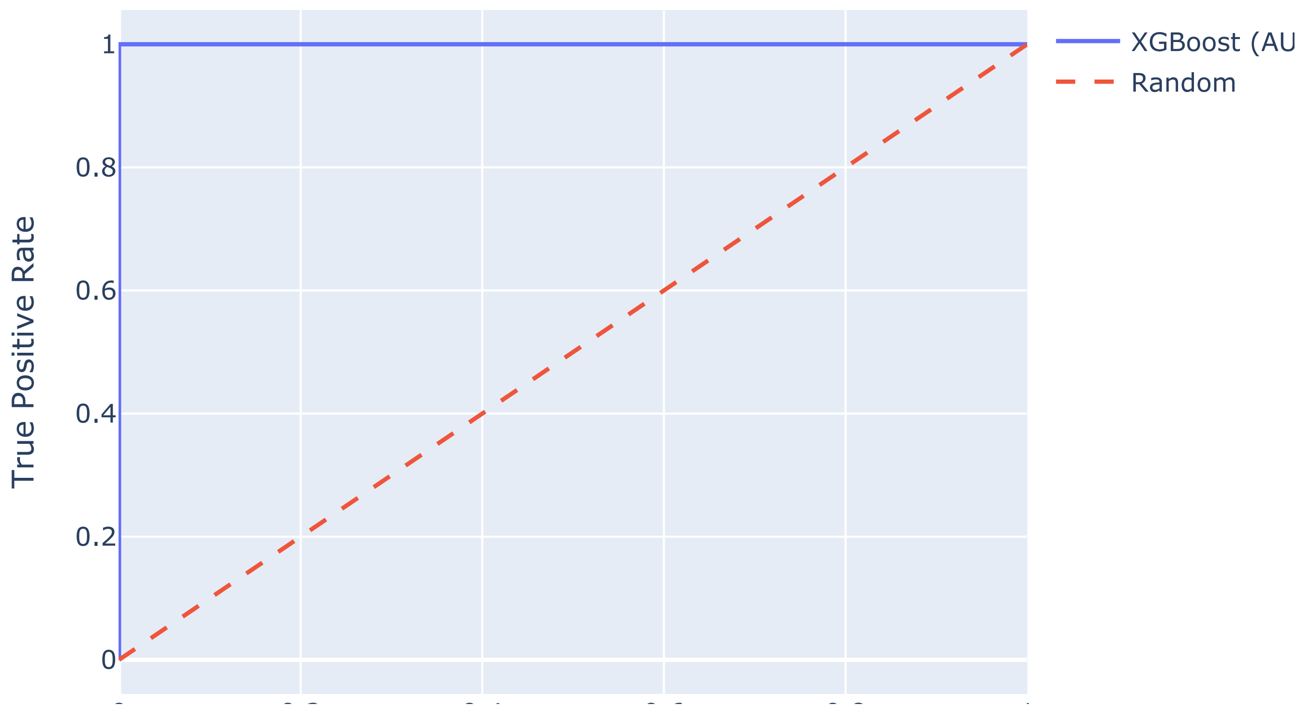


ROC Curve - Type_of_Food_Allergy_Fish - RandomForest

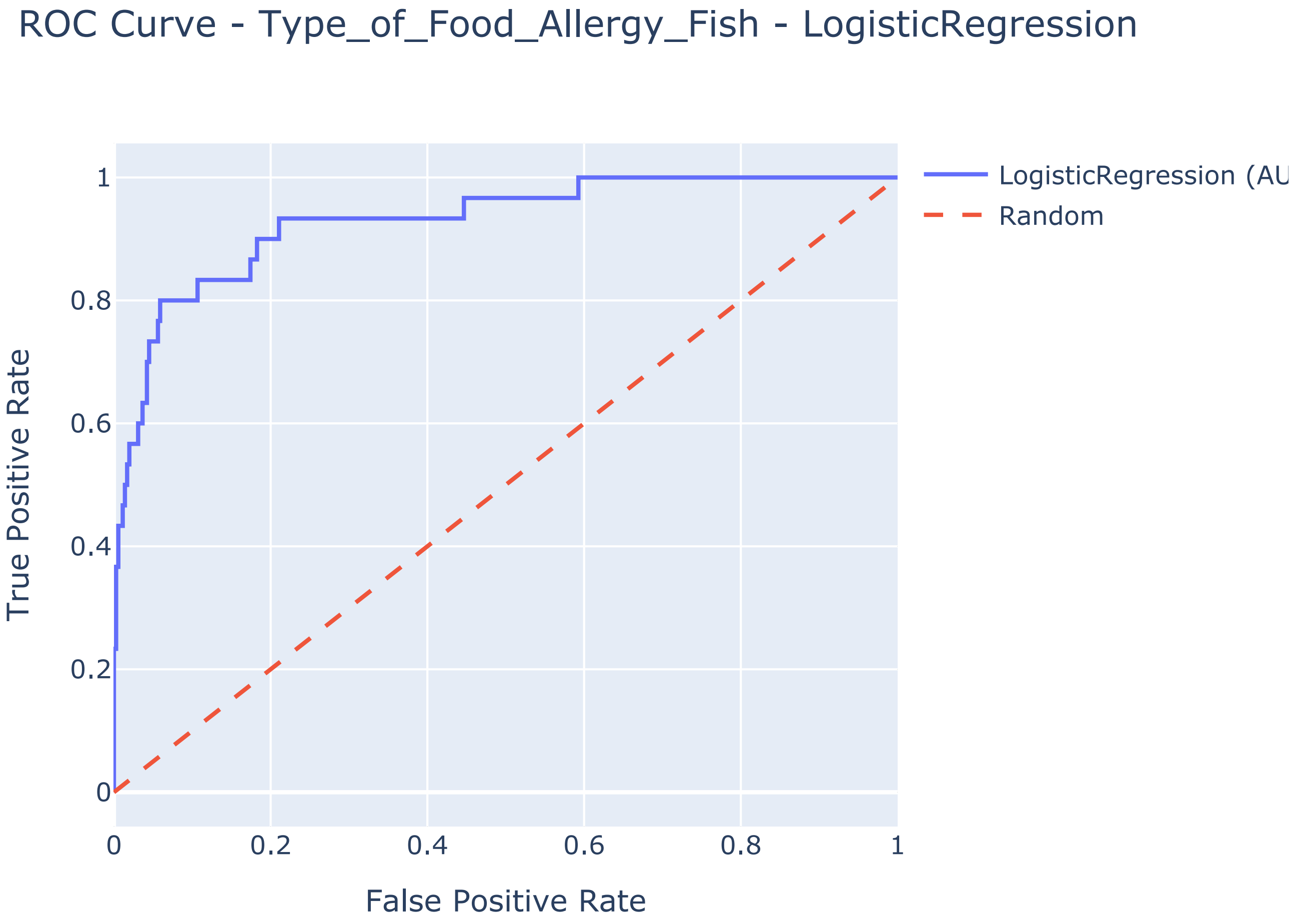


```
🔍 Target: Type_of_Food_Allergy_Fish | Model: XGBoost
📈 Accuracy: 0.9197
🎯 F1 (0): 0.9568 | F1 (1): 0.4152
📊 Precision: 0.9162 | AUC: 0.741031746031746
📊 Confusion Matrix:
 [[356   0]
 [  0  30]]
```

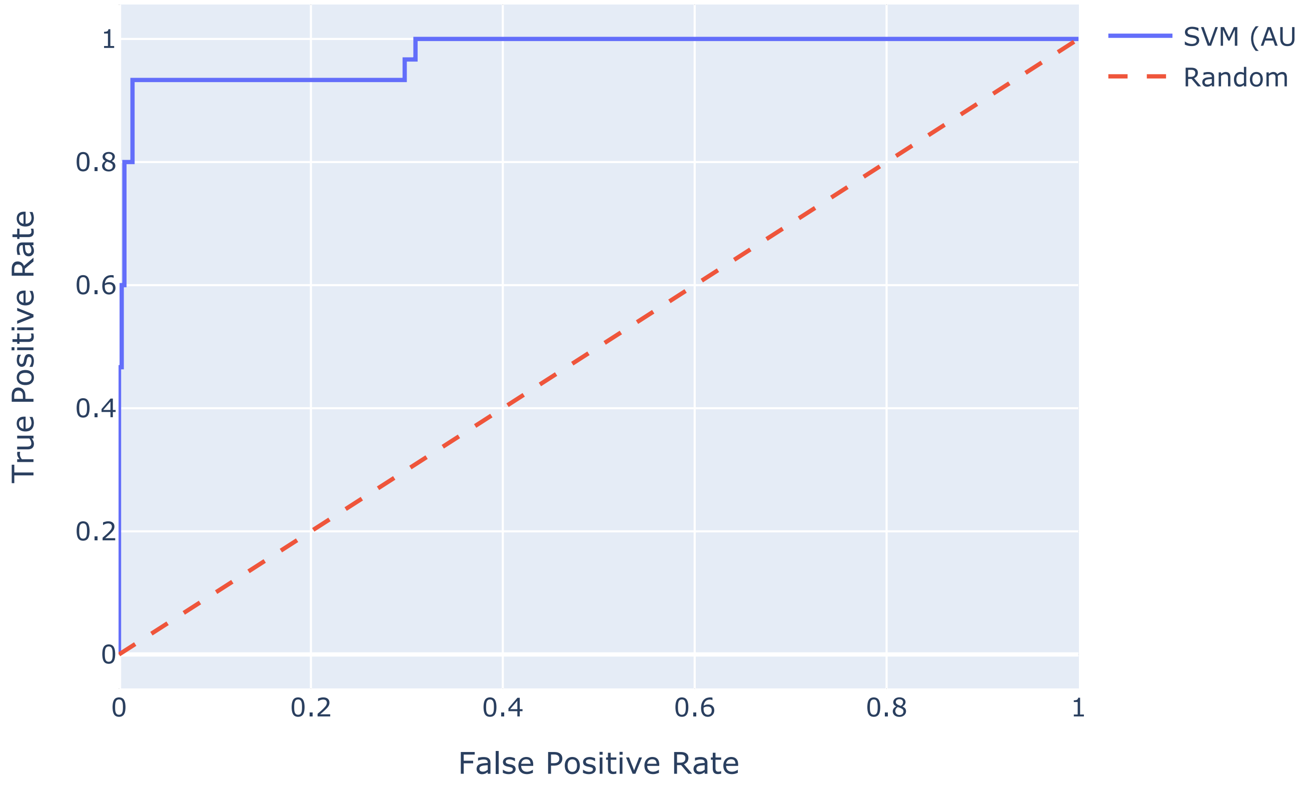
ROC Curve - Type_of_Food_Allergy_Fish - XGBoost



```
🔍 Target: Type_of_Food_Allergy_Fish | Model: LogisticRegression
📈 Accuracy: 0.8372
🎯 F1 (0): 0.9076 | F1 (1): 0.2415
📊 Precision: 0.8818 | AUC: 0.6347354497354497
📊 Confusion Matrix:
 [[355   1]
 [ 20  10]]
```

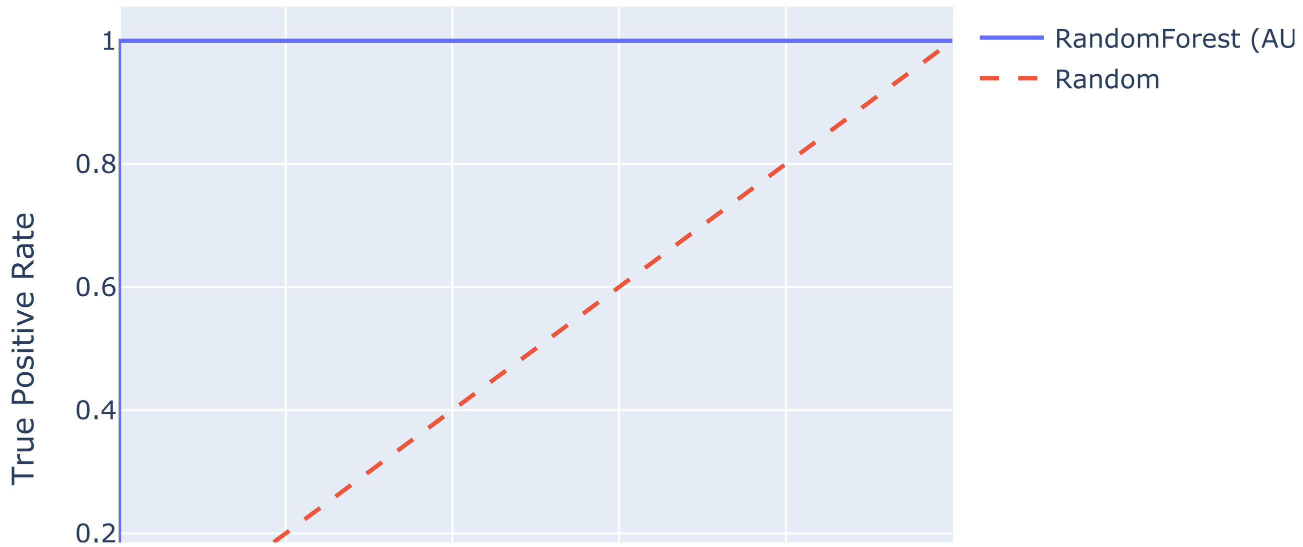


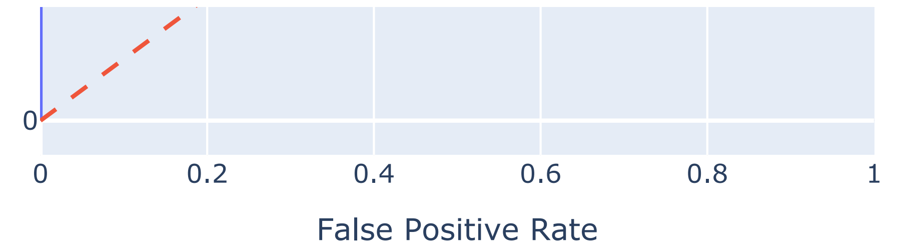
```
🔍 Target: Type_of_Food_Allergy_Fish | Model: SVM
📈 Accuracy: 0.7723
🎯 F1 (0): 0.8651 | F1 (1): 0.1530
📊 Precision: 0.8626 | AUC: 0.5847619047619047
📊 Confusion Matrix:
 [[356   0]
 [ 30   0]]
```



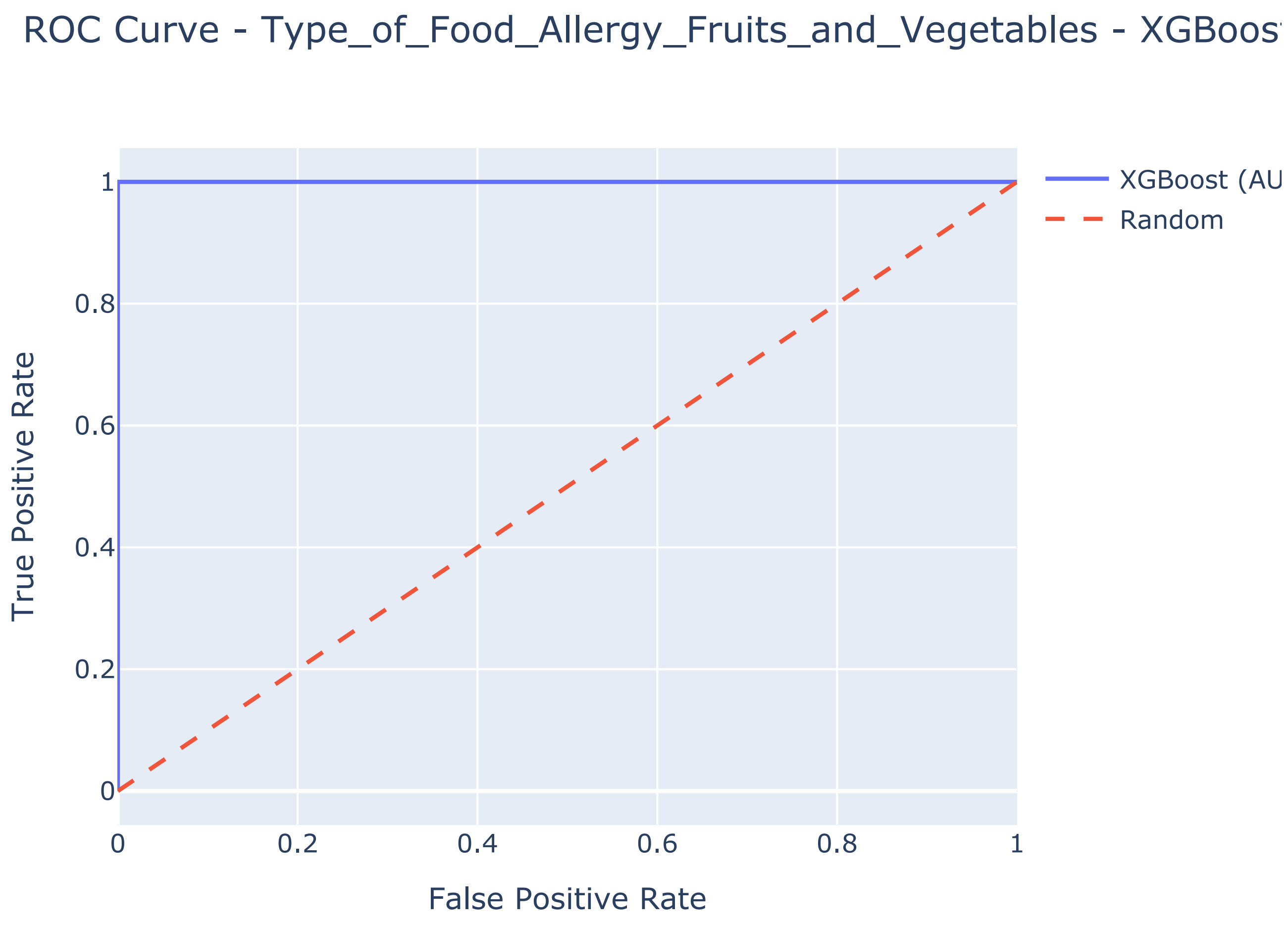
```
🔍 Target: Type_of_Food_Allergy_Fruits_and_Vegetables | Model: RandomForest
📈 Accuracy: 0.7980
🎯 F1 (0): 0.8826 | F1 (1): 0.2698
📊 Precision: 0.7634 | AUC: 0.6586633544546852
📊 Confusion Matrix:
 [[318   0]
 [  0  68]]
```

ROC Curve - Type_of_Food_Allergy_Fruits_and_Vegetables - Randor



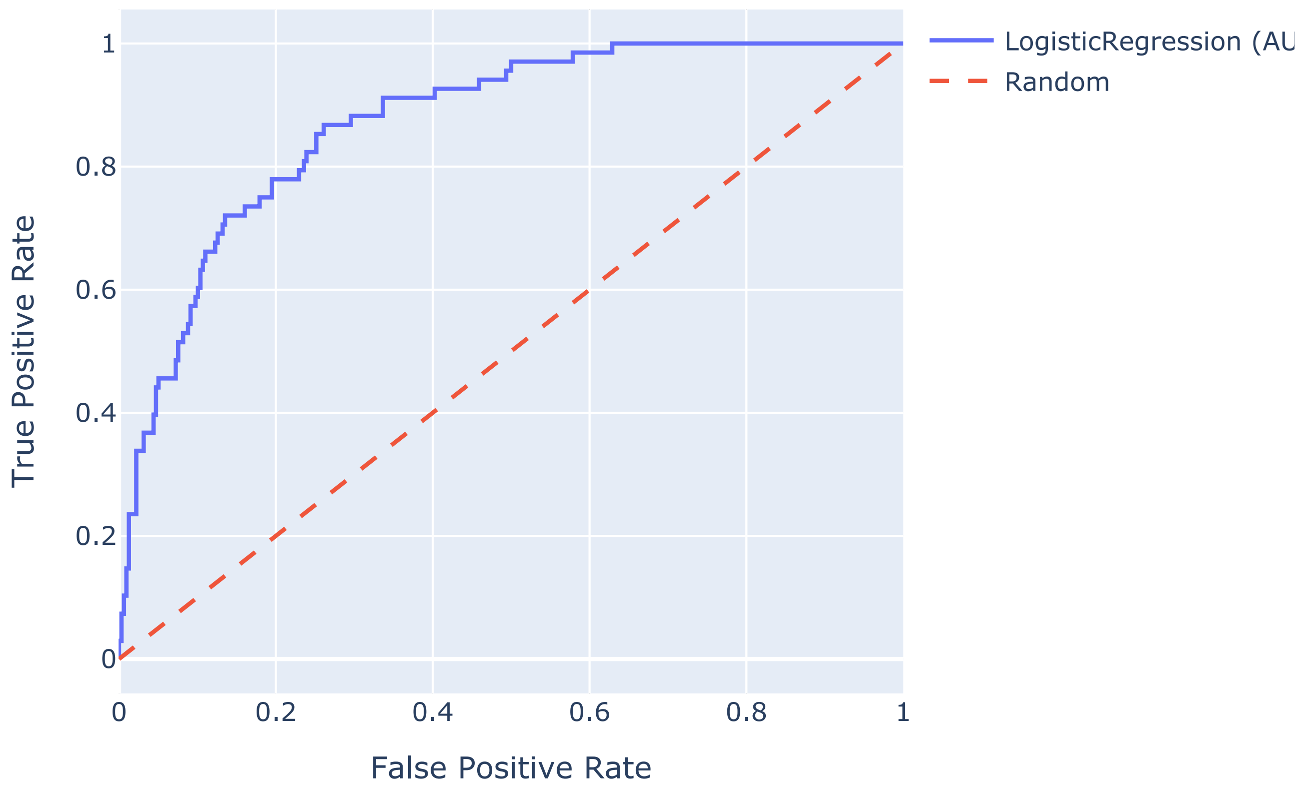


```
🔍 Target: Type_of_Food_Allergy_Fruits_and_Vegetables | Model: XGBoost
📈 Accuracy: 0.7900
🎯 F1 (0): 0.8766 | F1 (1): 0.2745
📊 Precision: 0.7627 | AUC: 0.6096078149001536
📊 Confusion Matrix:
 [[318   0]
 [  0  68]]
```



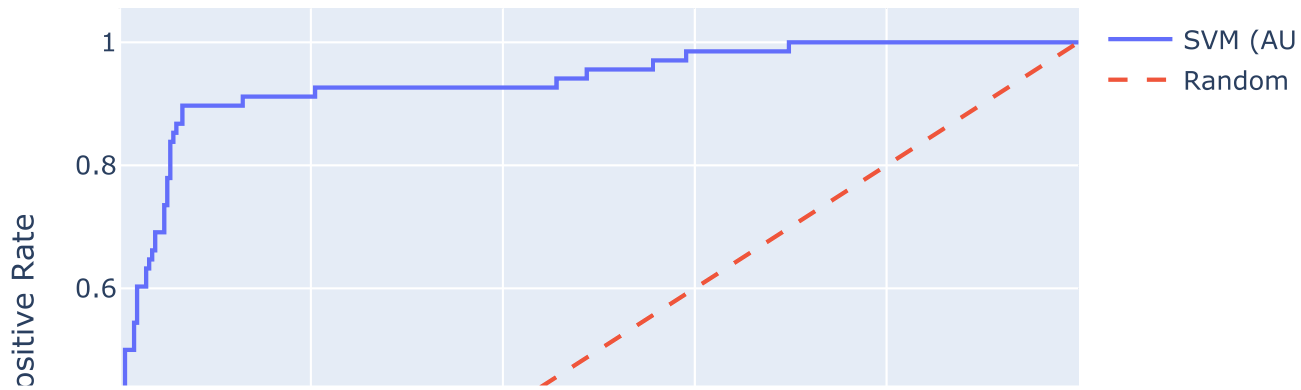
```
🔍 Target: Type_of_Food_Allergy_Fruits_and_Vegetables | Model: LogisticRegr
📈 Accuracy: 0.7276
🎯 F1 (0): 0.8314 | F1 (1): 0.2570
📊 Precision: 0.7404 | AUC: 0.628168202764977
📊 Confusion Matrix:
 [[311   7]
 [ 48  20]]
```

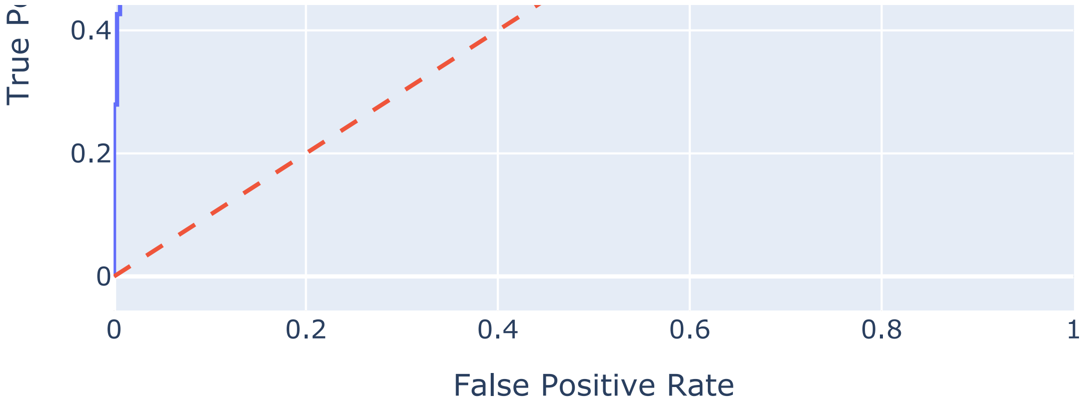
ROC Curve - Type_of_Food_Allergy_Fruits_and_Vegetables - Logisticl



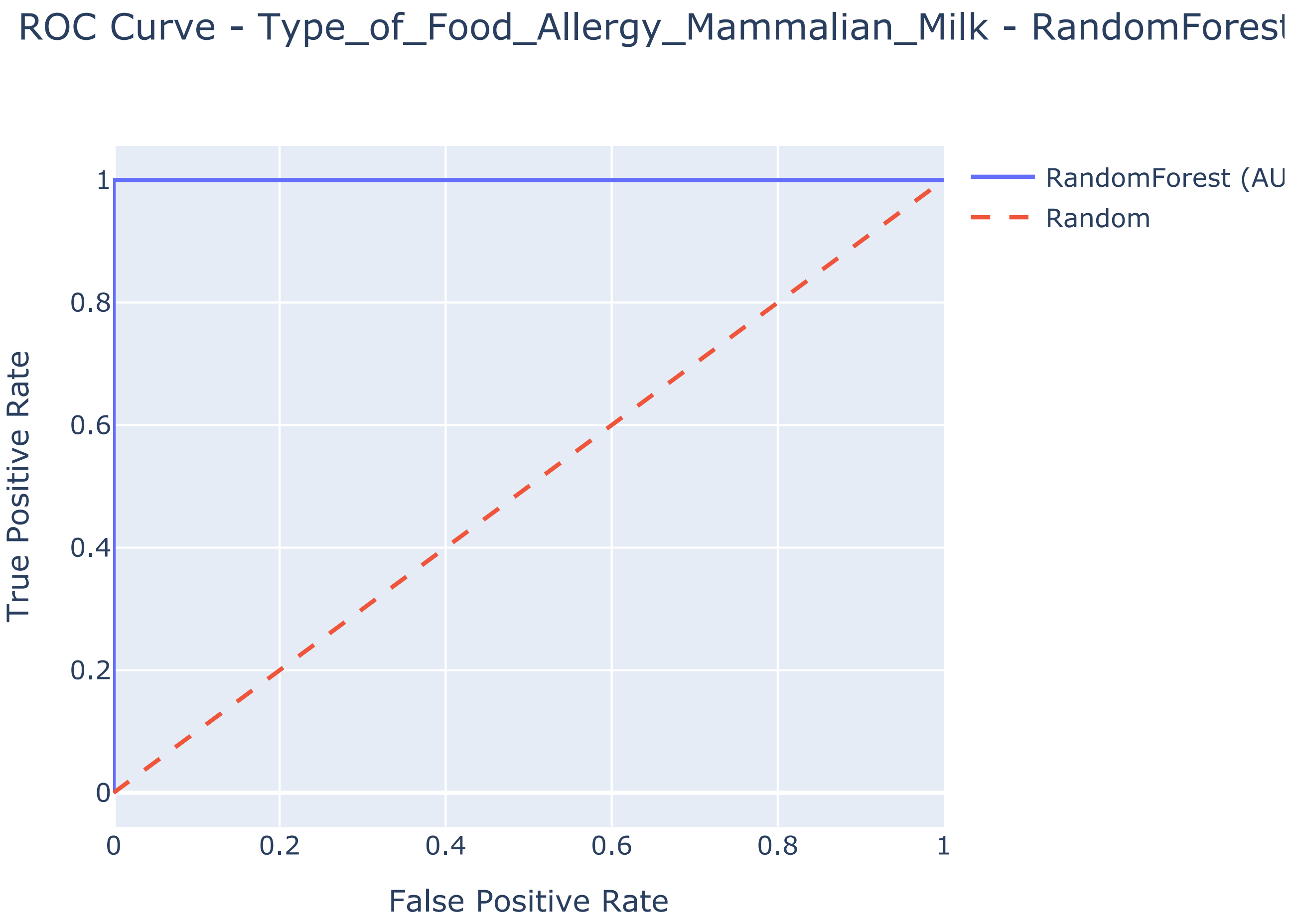
```
🔍 Target: Type_of_Food_Allergy_Fruits_and_Vegetables | Model: SVM
📈 Accuracy: 0.6192
🎯 F1 (0): 0.7269 | F1 (1): 0.3523
📊 Precision: 0.7689 | AUC: 0.628312211981567
📊 Confusion Matrix:
 [[318   0]
 [ 68   0]]
```

ROC Curve - Type_of_Food_Allergy_Fruits_and_Vegetables - SVM



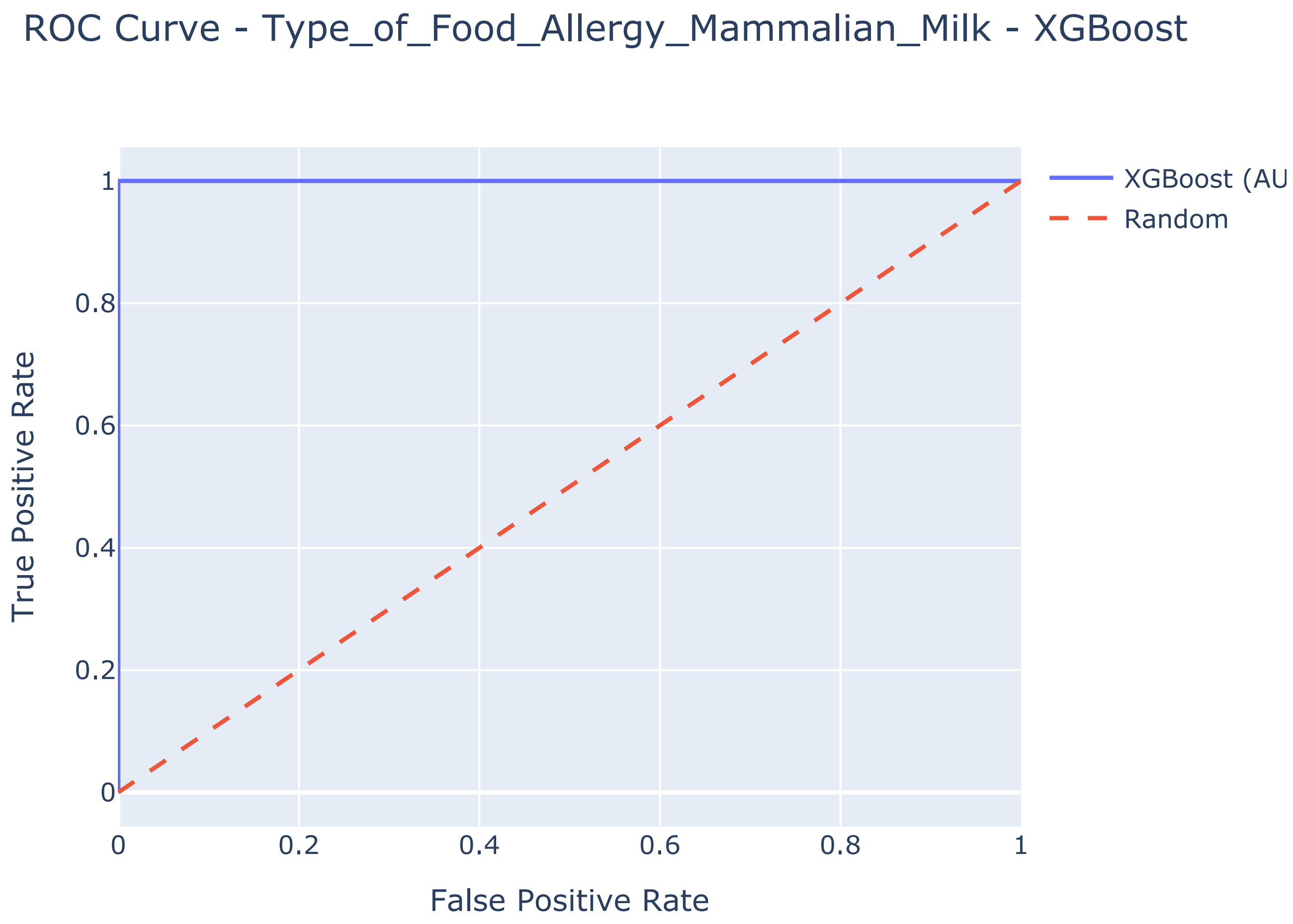


```
🔍 Target: Type_of_Food_Allergy_Mammalian_Milk | Model: RandomForest
📈 Accuracy: 0.9429
🎯 F1 (0): 0.9702 | F1 (1): 0.2333
📊 Precision: 0.9210 | AUC: 0.7624687187187188
📊 Confusion Matrix:
 [[364   0]
 [  0  22]]
```

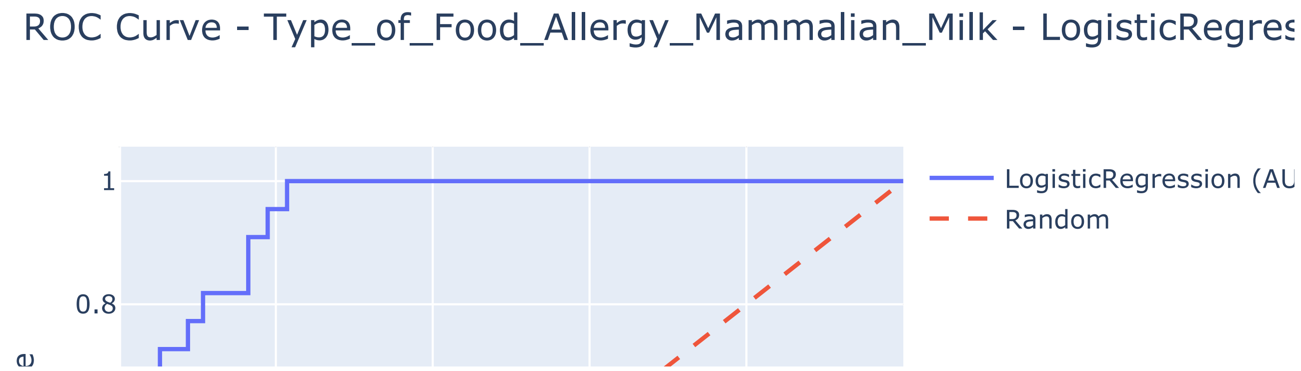


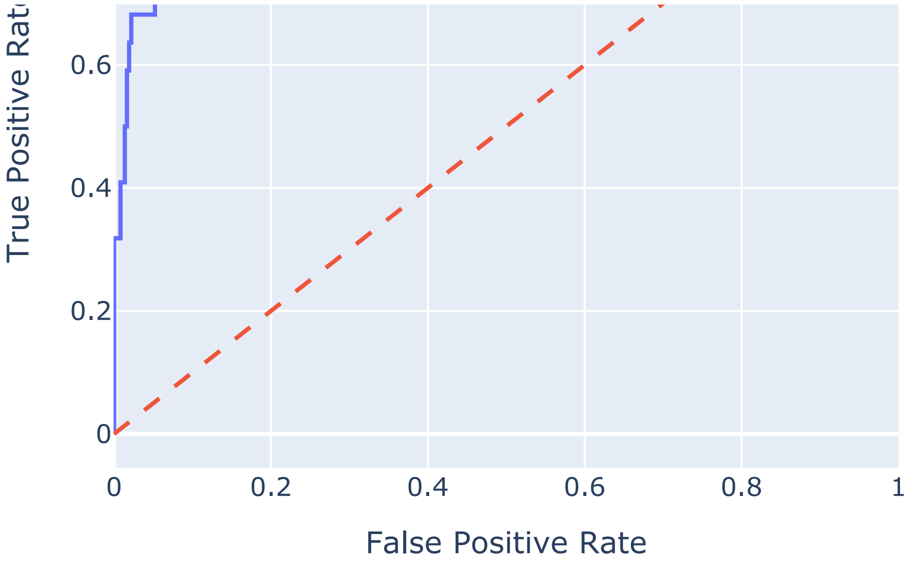
```
🔍 Target: Type_of_Food_Allergy_Mammalian_Milk | Model: XGBoost
📈 Accuracy: 0.9221
```

```
🎯 F1 (0): 0.9584 | F1 (1): 0.2805
📊 Precision: 0.9233 | AUC: 0.7304429429429429
📊 Confusion Matrix:
 [[364   0]
 [  0  22]]
```

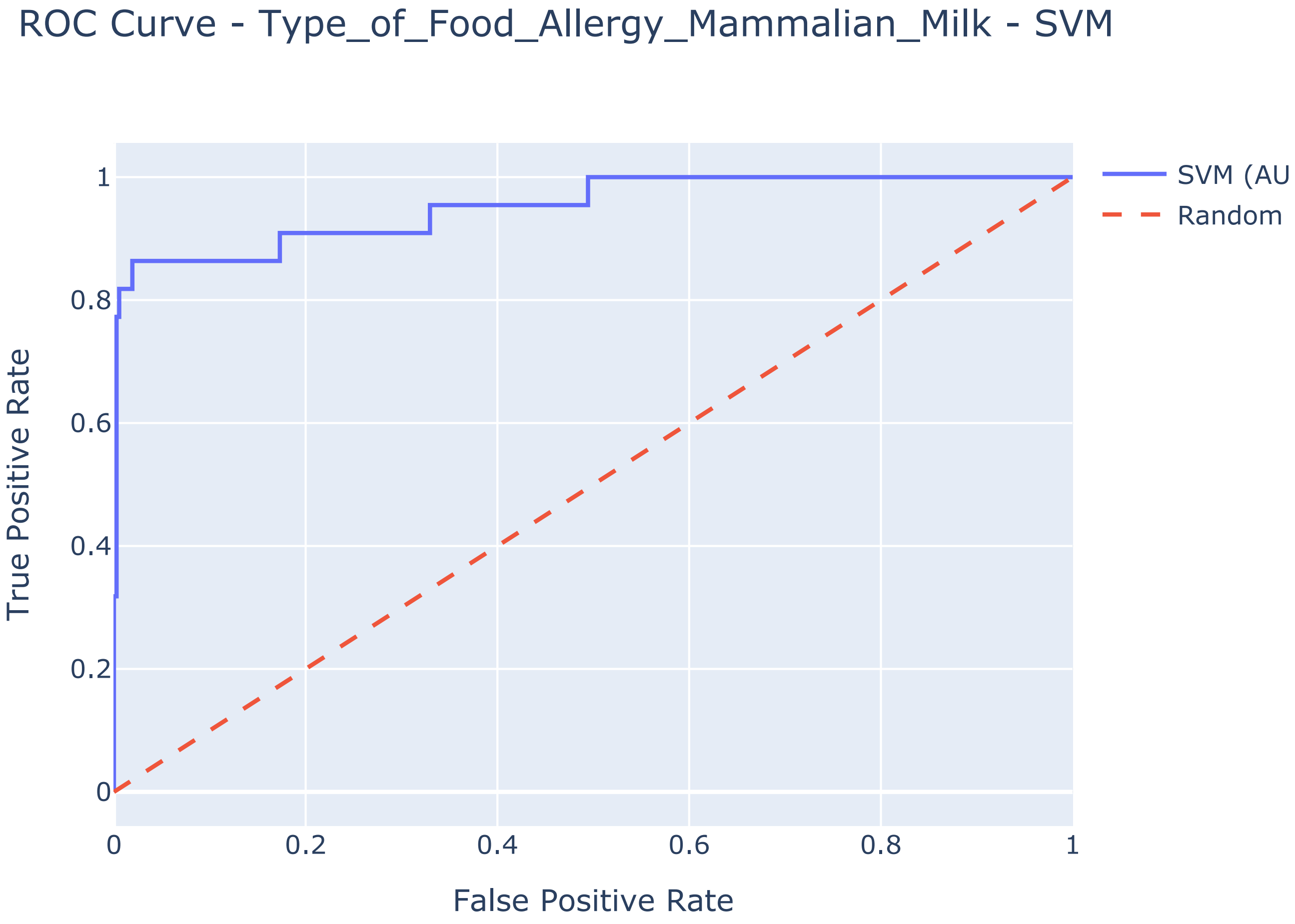

ROC Curve - Type_of_Food_Allergy_Mammalian_Milk - XGBoost

```
🔍 Target: Type_of_Food_Allergy_Mammalian_Milk | Model: LogisticRegression
📈 Accuracy: 0.8857
🎯 F1 (0): 0.9382 | F1 (1): 0.2319
📊 Precision: 0.9165 | AUC: 0.6832332332332333
📊 Confusion Matrix:
 [[362   2]
 [ 15   7]]
```


ROC Curve - Type_of_Food_Allergy_Mammalian_Milk - LogisticRegres

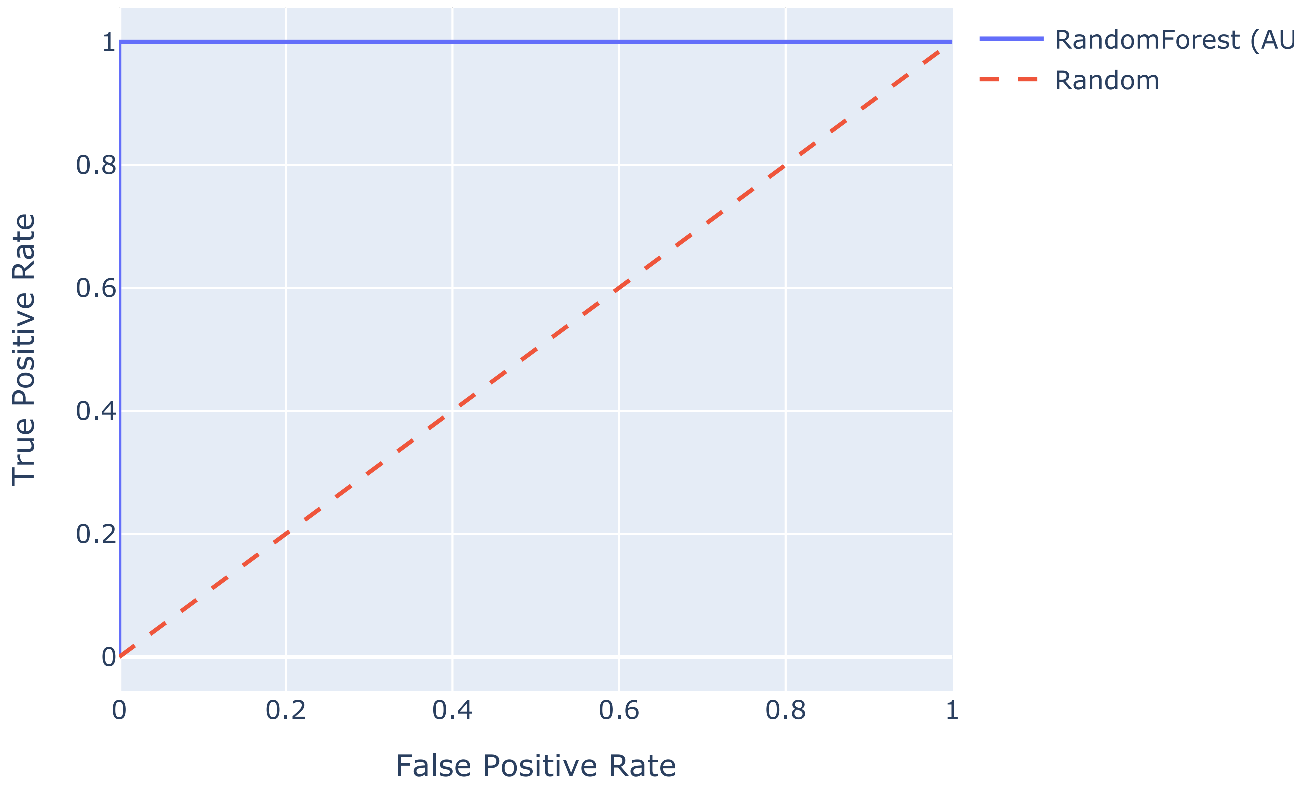


```
🔍 Target: Type_of_Food_Allergy_Mammalian_Milk | Model: SVM
📈 Accuracy: 0.7999
🎯 F1 (0): 0.8860 | F1 (1): 0.1208
📊 Precision: 0.8983 | AUC: 0.5212962962962963
📊 Confusion Matrix:
 [[364   0]
 [ 22   0]]
```



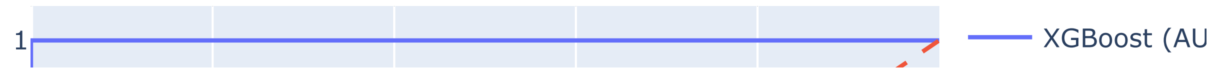
```
🔍 Target: Type_of_Food_Allergy_Oral_Syndrom | Model: RandomForest
📈 Accuracy: 0.9636
🎯 F1 (0): 0.9803 | F1 (1): 0.7588
📊 Precision: 0.9655 | AUC: 0.9905777310924371
📊 Confusion Matrix:
 [[348   0]
 [  0  38]]
```

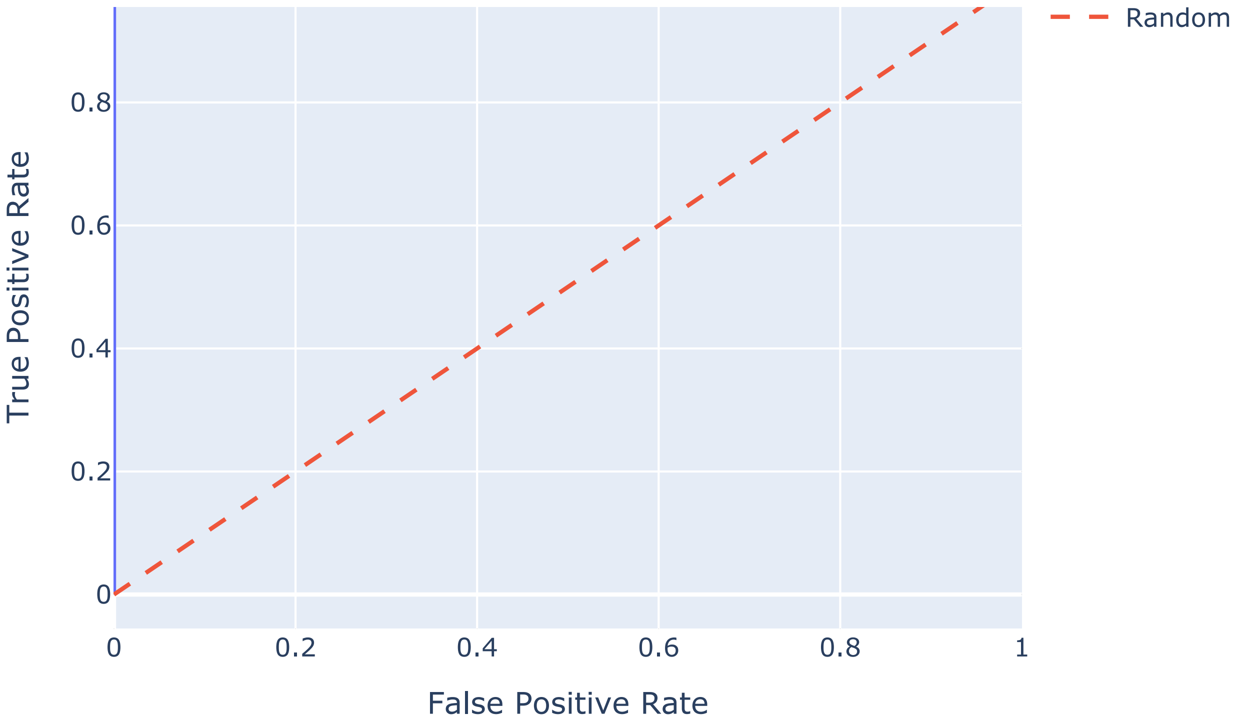
ROC Curve - Type_of_Food_Allergy_Oral_Syndrom - RandomForest



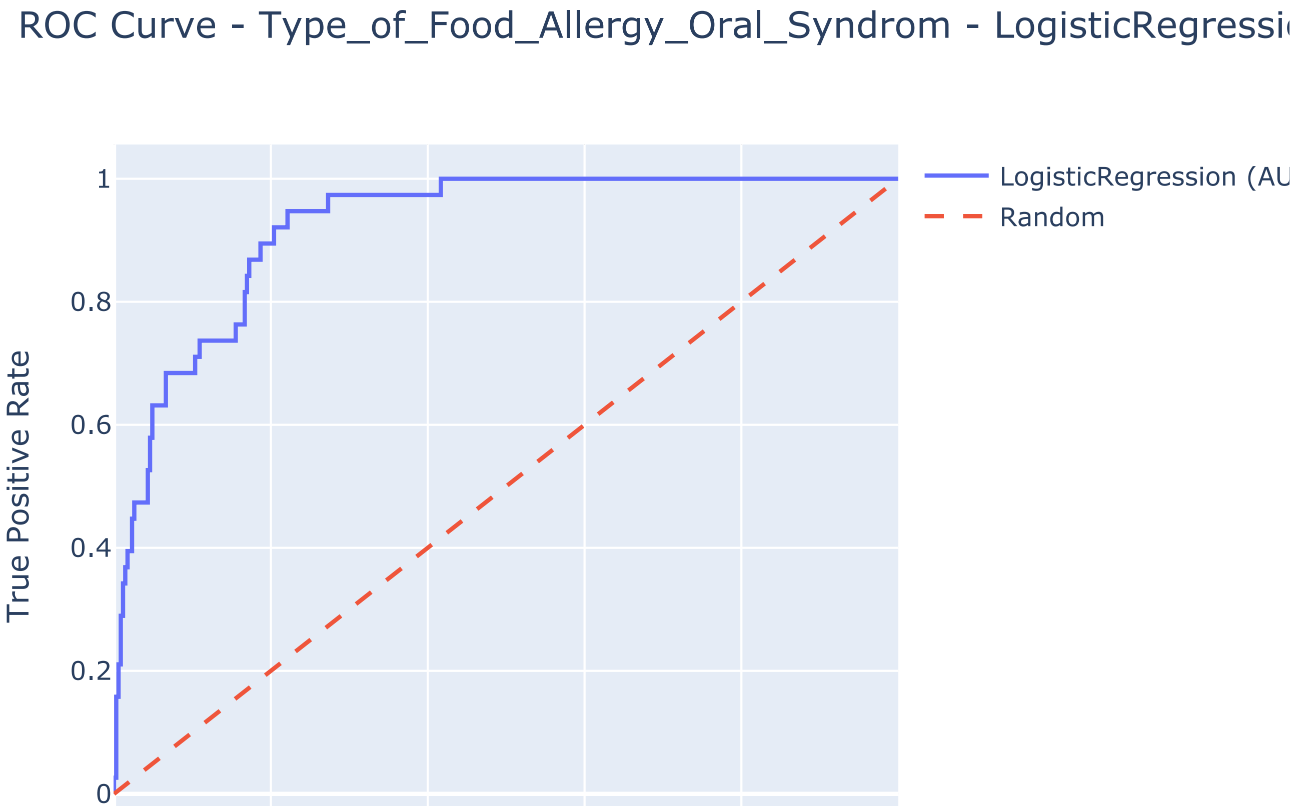
```
🔍 Target: Type_of_Food_Allergy_Oral_Syndrom | Model: XGBoost
📈 Accuracy: 1.0000
🎯 F1 (0): 1.0000 | F1 (1): 1.0000
📊 Precision: 1.0000 | AUC: 1.0
📊 Confusion Matrix:
 [[348   0]
 [  0  38]]
```

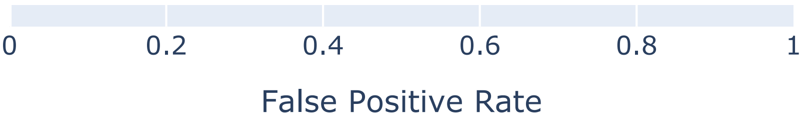
ROC Curve - Type_of_Food_Allergy_Oral_Syndrom - XGBoost





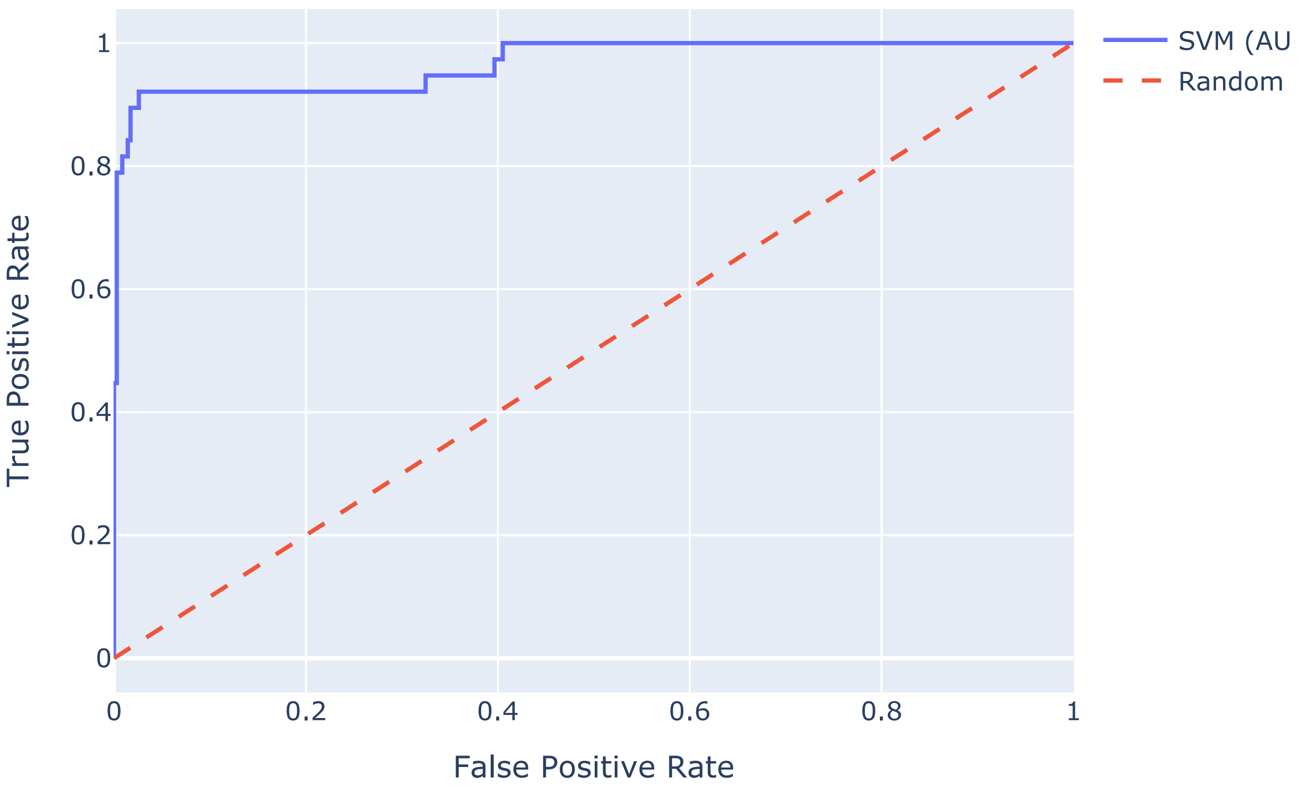
```
🔍 Target: Type_of_Food_Allergy_Oral_Syndrom | Model: LogisticRegression
📈 Accuracy: 0.8165
🎯 F1 (0): 0.8940 | F1 (1): 0.2370
📊 Precision: 0.8508 | AUC: 0.6590756302521009
📊 Confusion Matrix:
 [[345   3]
 [ 28  10]]
```





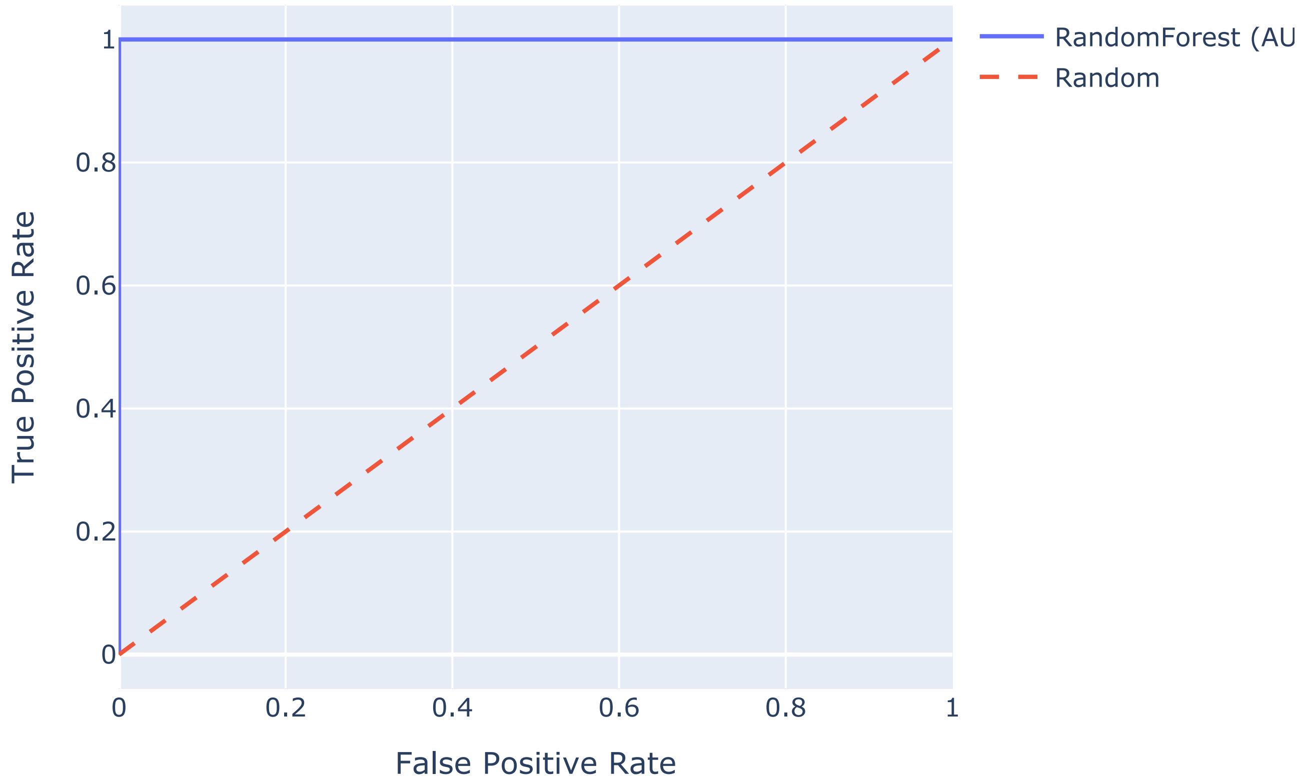
```
🔍 Target: Type_of_Food_Allergy_Oral_Syndrom | Model: SVM
📈 Accuracy: 0.5959
🎯 F1 (0): 0.7268 | F1 (1): 0.1881
📊 Precision: 0.8421 | AUC: 0.5966666666666667
📊 Confusion Matrix:
 [[348   0]
 [ 38   0]]
```

ROC Curve - Type_of_Food_Allergy_Oral_Syndrom - SVM



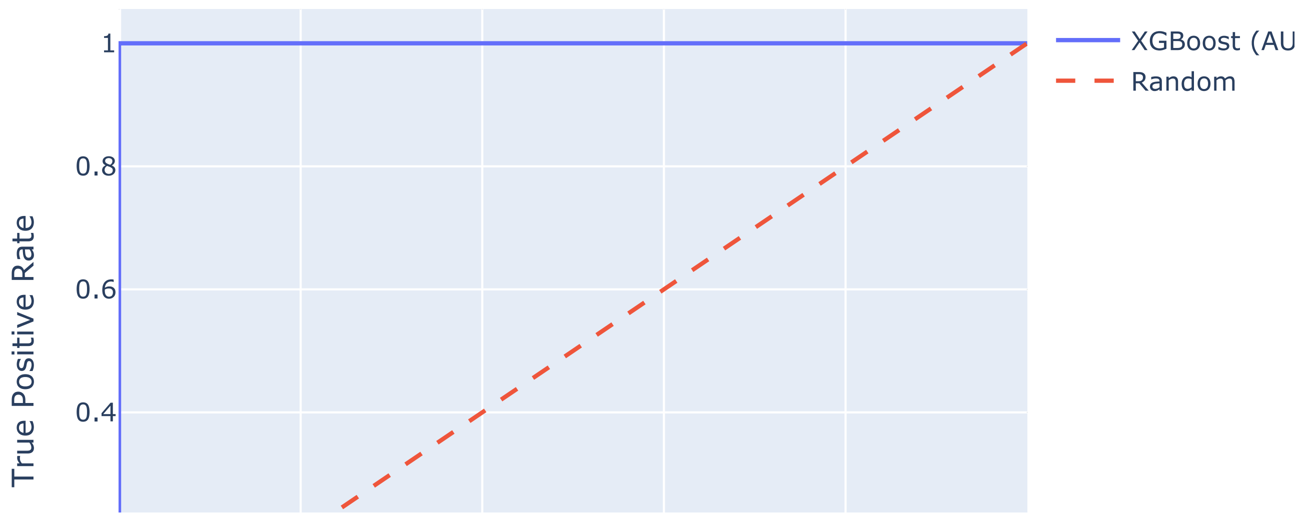
```
🔍 Target: Type_of_Food_Allergy_Other_Legumes | Model: RandomForest
📈 Accuracy: 0.9013
🎯 F1 (0): 0.9479 | F1 (1): 0.0000
📊 Precision: 0.8490 | AUC: 0.6398412698412699
📊 Confusion Matrix:
 [[356   0]
 [  0  30]]
```

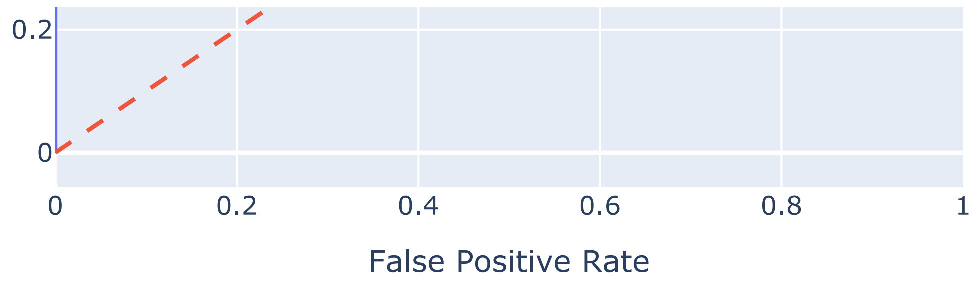
ROC Curve - Type_of_Food_Allergy_Other_Legumes - RandomForest



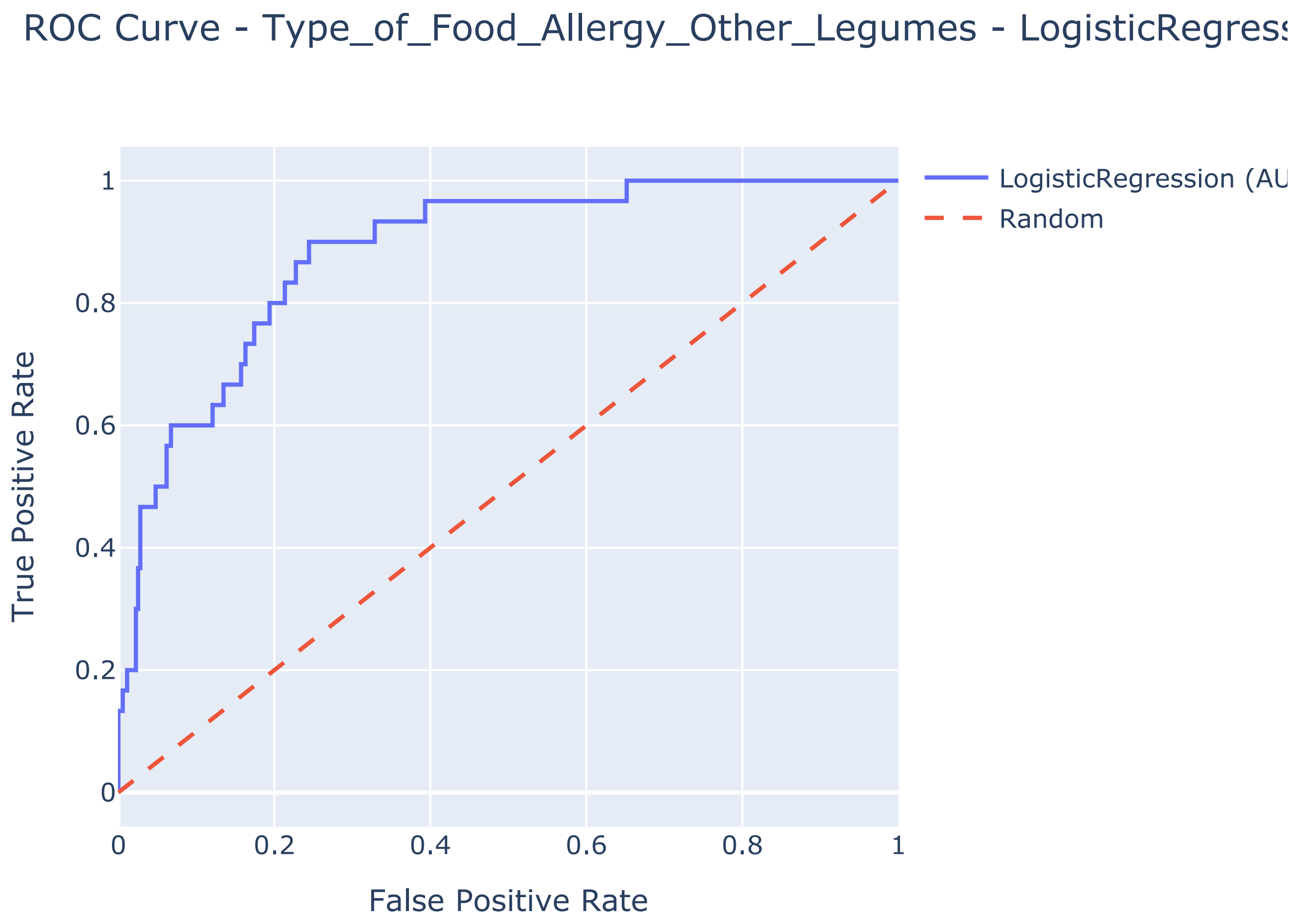
```
🔍 Target: Type_of_Food_Allergy_Other_Legumes | Model: XGBoost
📈 Accuracy: 0.8860
🎯 F1 (0): 0.9391 | F1 (1): 0.0667
📊 Precision: 0.8574 | AUC: 0.5533862433862434
📊 Confusion Matrix:
 [[356   0]
 [  0  30]]
```

ROC Curve - Type_of_Food_Allergy_Other_Legumes - XGBoost

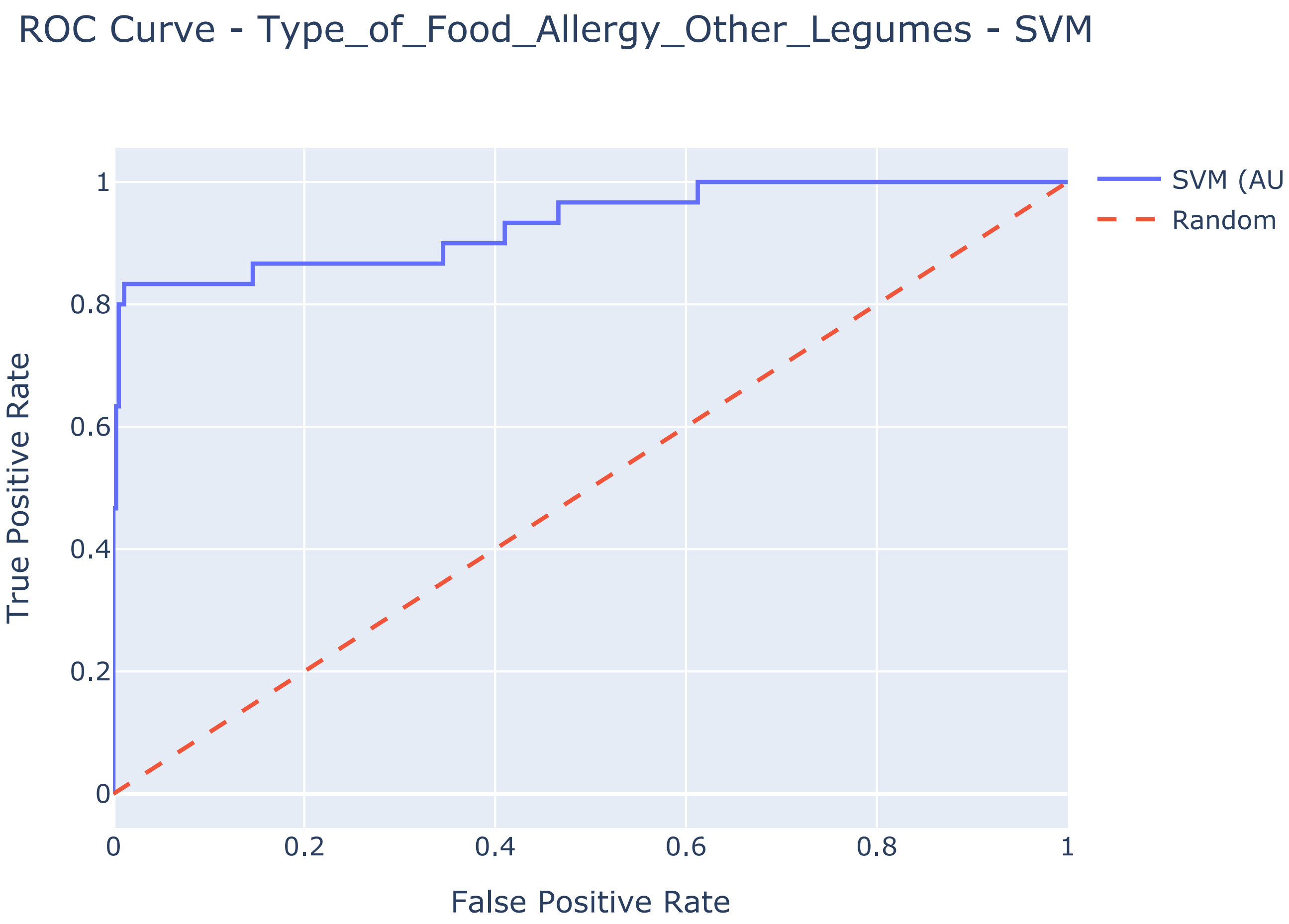




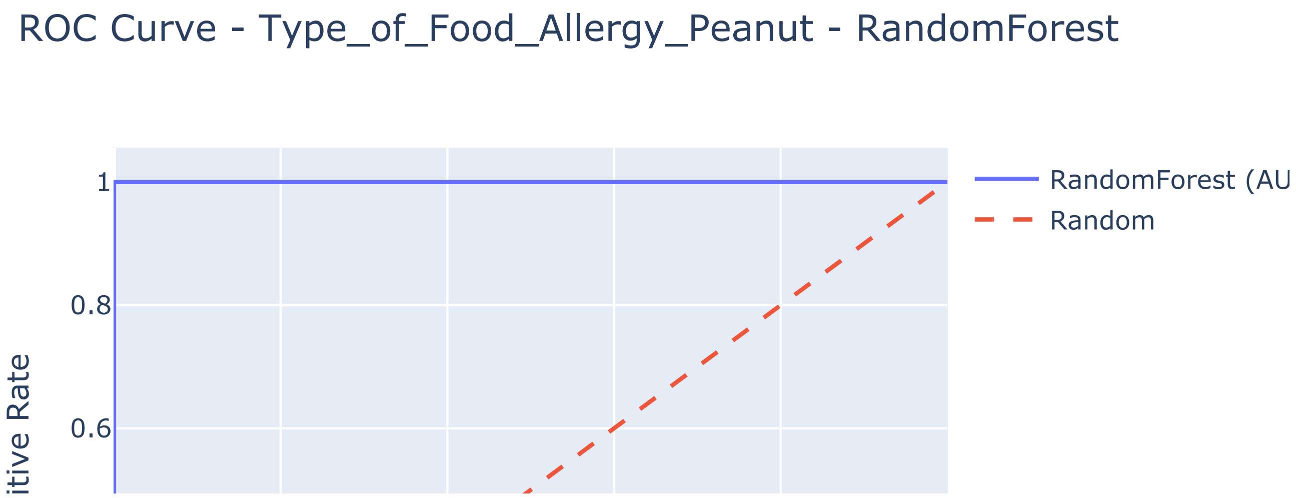
```
🔍 Target: Type_of_Food_Allergy_Other_Legumes | Model: LogisticRegression
📈 Accuracy: 0.8163
🎯 F1 (0): 0.8972 | F1 (1): 0.0708
📊 Precision: 0.8532 | AUC: 0.4314550264550264
📊 Confusion Matrix:
 [[356   0]
 [ 29   1]]
```



```
🔍 Target: Type_of_Food_Allergy_Other_Legumes | Model: SVM
📈 Accuracy: 0.6480
🎯 F1 (0): 0.7768 | F1 (1): 0.1298
📊 Precision: 0.8570 | AUC: 0.4792328042328043
📊 Confusion Matrix:
 [[356   0]
 [ 30    0]]
```



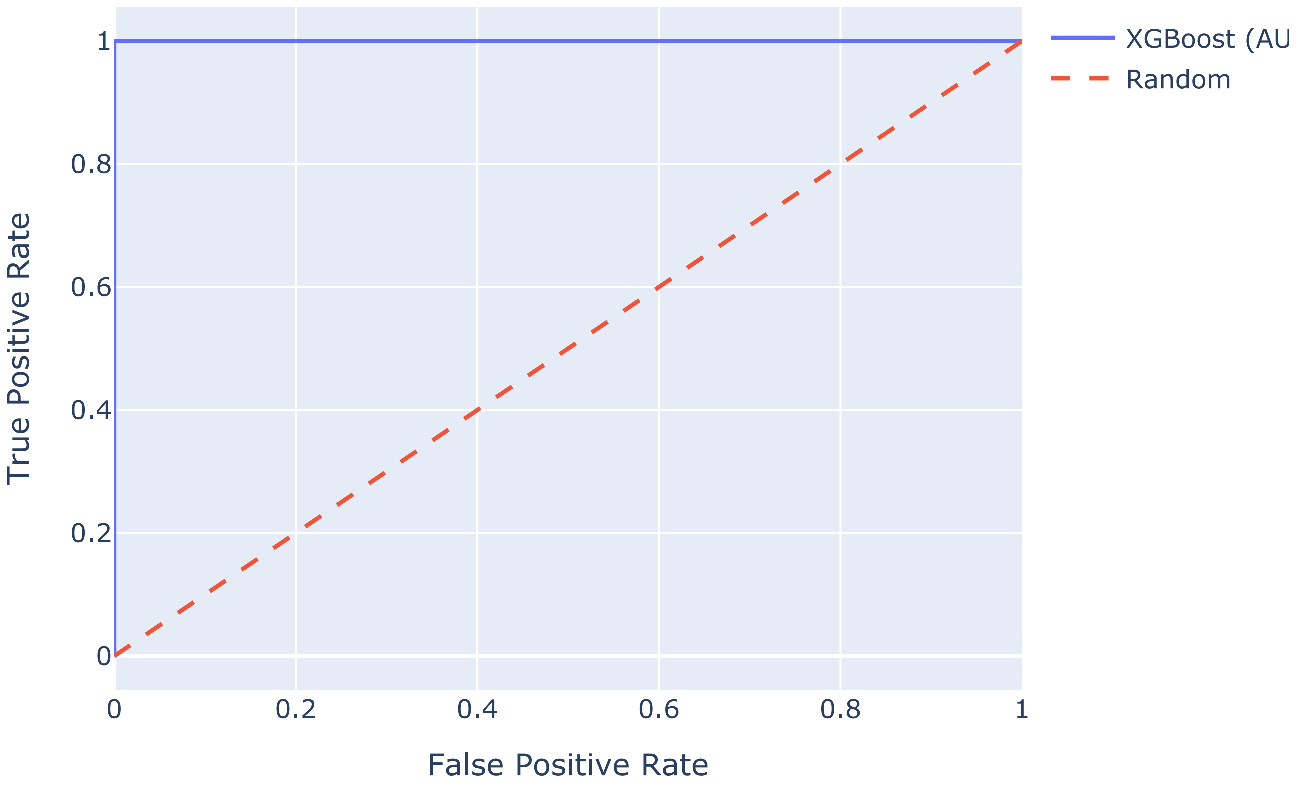
```
🔍 Target: Type_of_Food_Allergy_Peanut | Model: RandomForest
📈 Accuracy: 0.8060
🎯 F1 (0): 0.8891 | F1 (1): 0.1999
📊 Precision: 0.7751 | AUC: 0.6600047348484848
📊 Confusion Matrix:
 [[329   0]
 [  0  57]]
```



True Pos
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
False Positive Rate

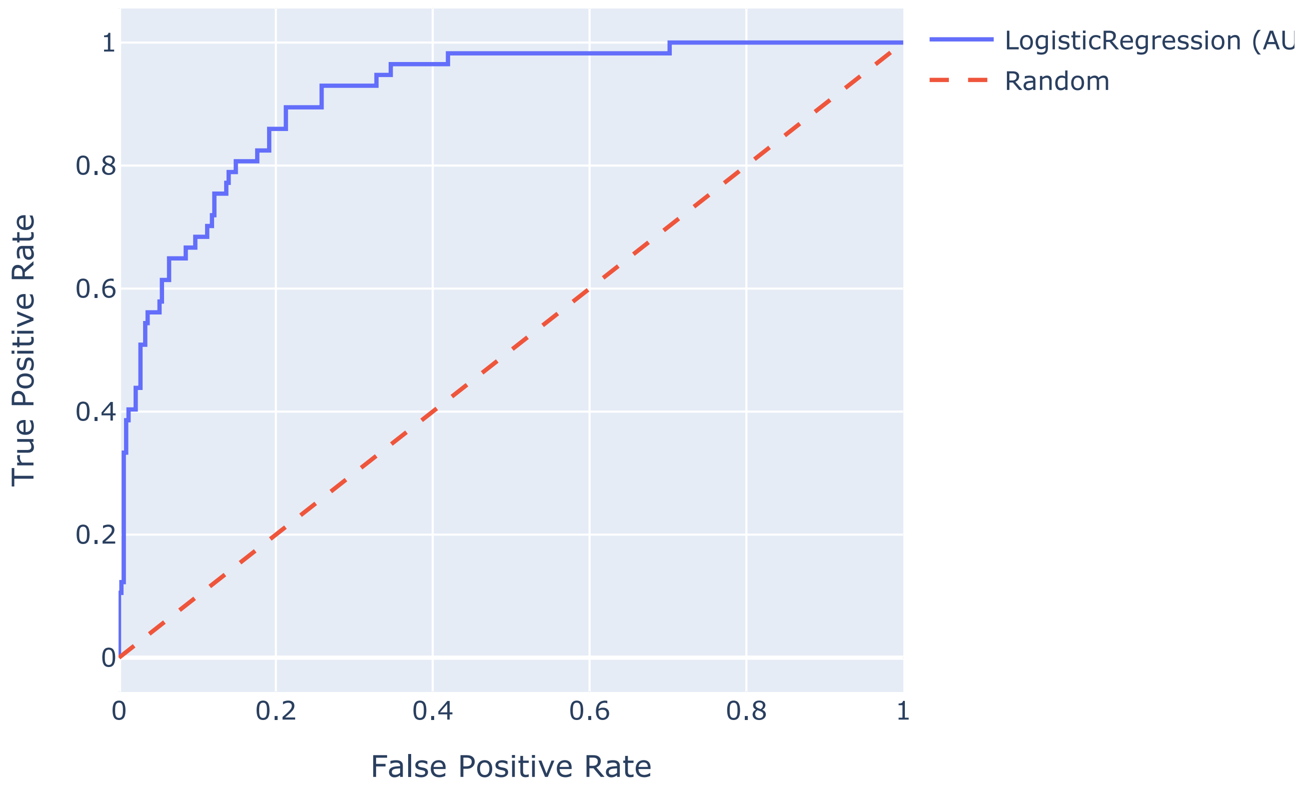
```
🔍 Target: Type_of_Food_Allergy_Peanut | Model: XGBoost
📈 Accuracy: 0.7981
🎯 F1 (0): 0.8841 | F1 (1): 0.1577
📊 Precision: 0.7622 | AUC: 0.6782575757575758
📊 Confusion Matrix:
 [[329   0]
 [  0  57]]
```

ROC Curve - Type_of_Food_Allergy_Peanut - XGBoost



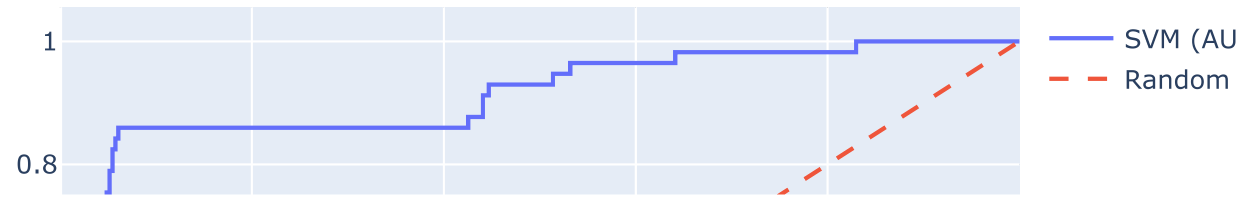
```
🔍 Target: Type_of_Food_Allergy_Peanut | Model: LogisticRegression
📈 Accuracy: 0.7589
📈 Accuracy: 0.7589
🎯 F1 (0): 0.8551 | F1 (1): 0.2661
📊 Precision: 0.7821 | AUC: 0.6446306818181818
📊 Confusion Matrix:
 [[326   3]
 [ 36  21]]
```

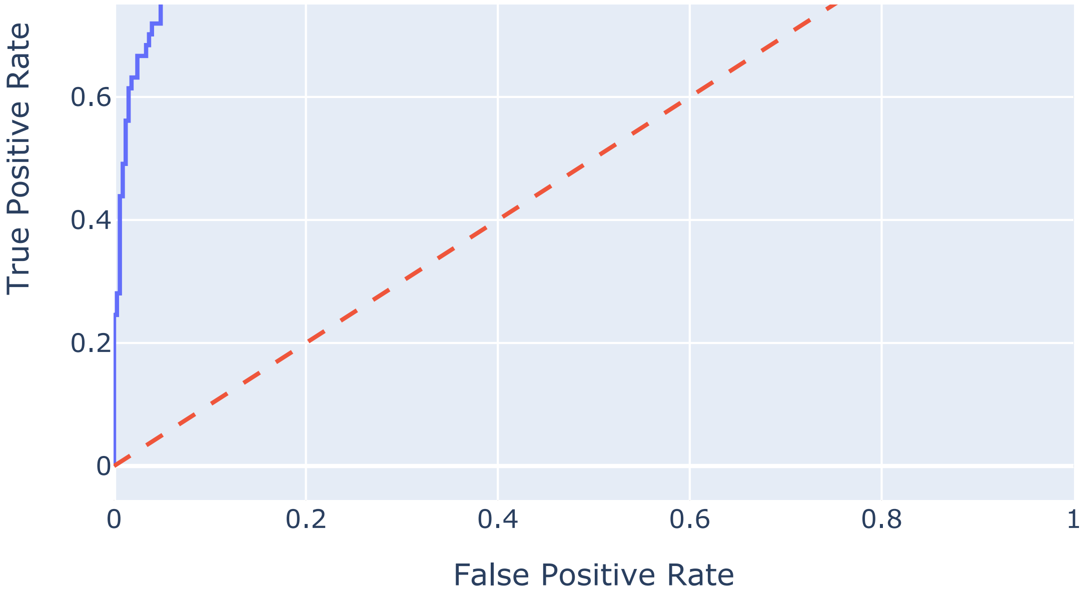
ROC Curve - Type_of_Food_Allergy_Peanut - LogisticRegression



```
🔍 Target: Type_of_Food_Allergy_Peanut | Model: SVM
📈 Accuracy: 0.7049
🎯 F1 (0): 0.8140 | F1 (1): 0.2527
📊 Precision: 0.7750 | AUC: 0.5907765151515151
📊 Confusion Matrix:
 [[329   0]
 [ 57   0]]
```

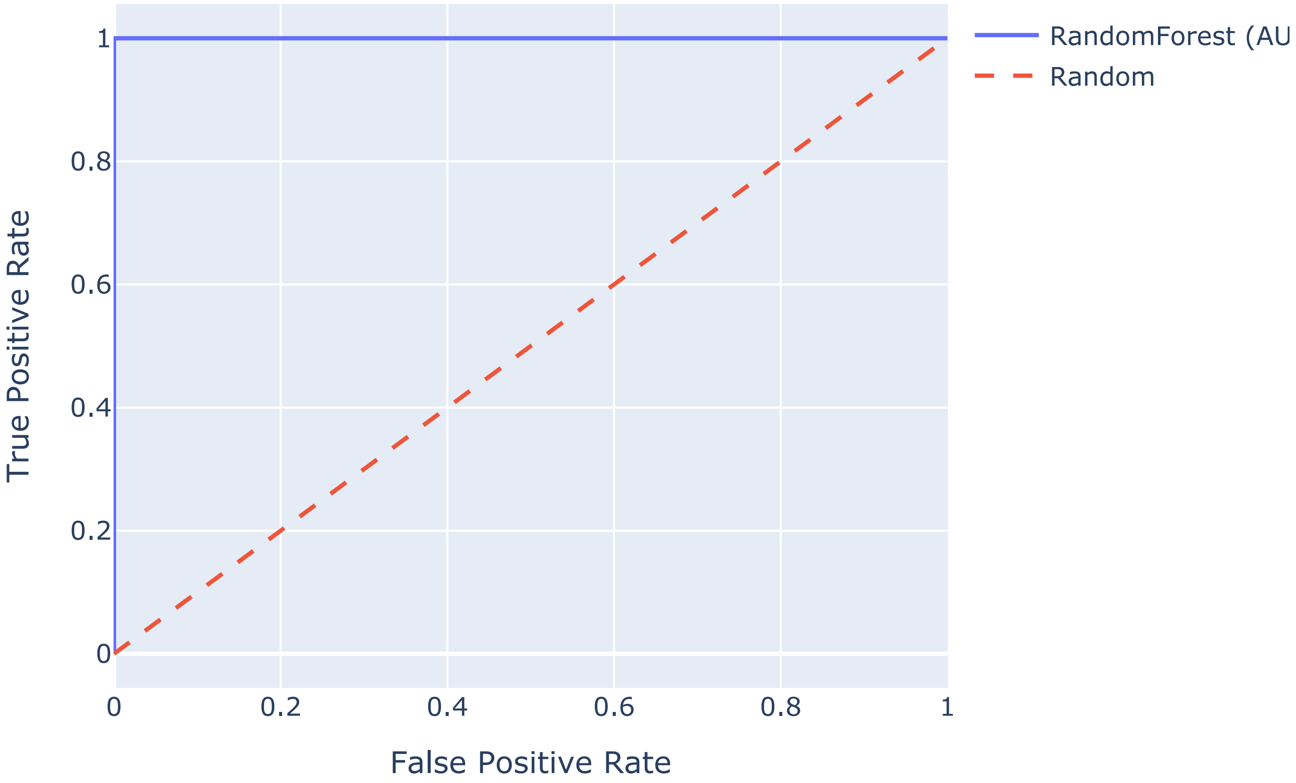
ROC Curve - Type_of_Food_Allergy_Peanut - SVM



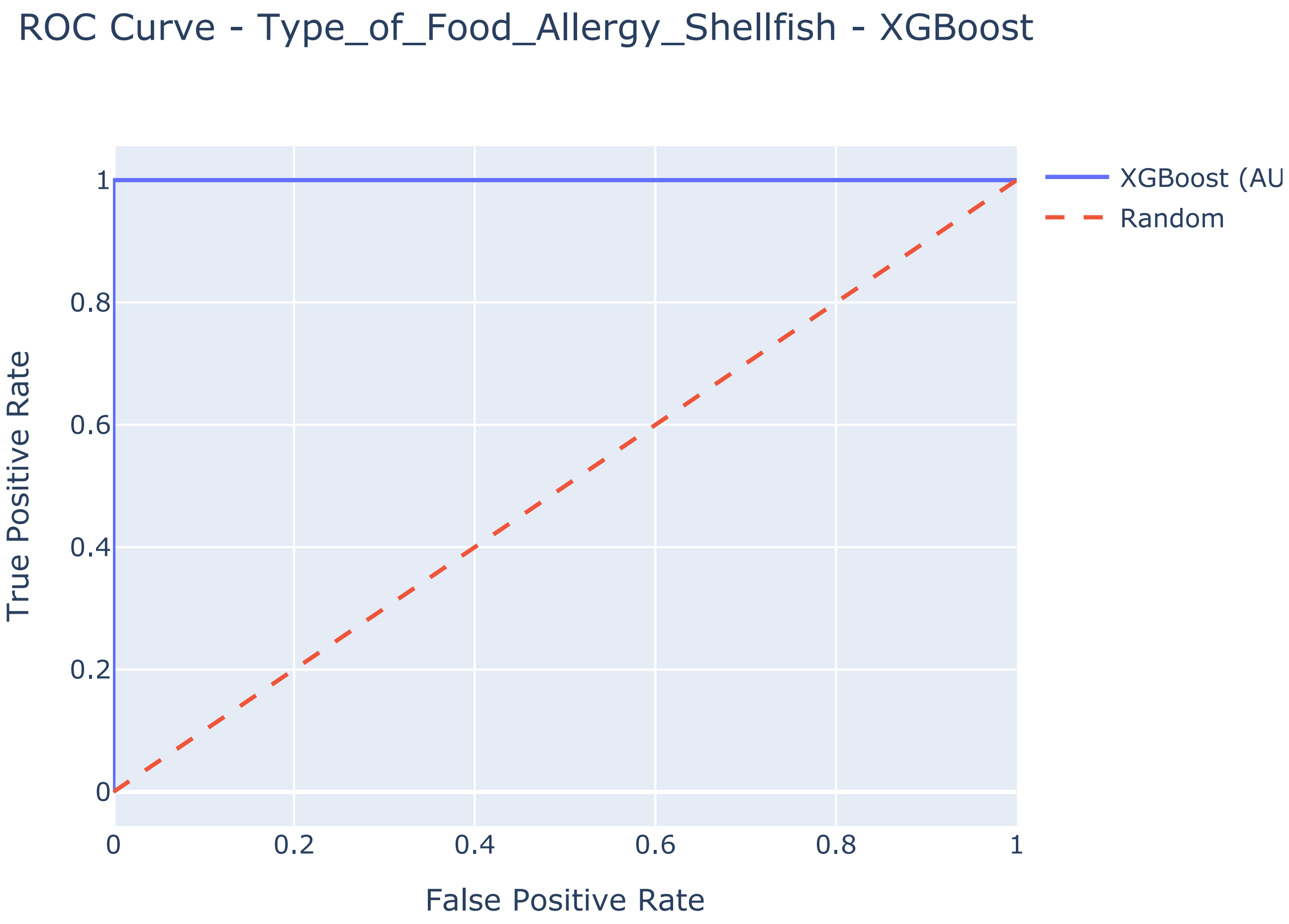


```
🔍 Target: Type_of_Food_Allergy_Shellfish | Model: RandomForest
📈 Accuracy: 0.9223
🎯 F1 (0): 0.9594 | F1 (1): 0.0500
📊 Precision: 0.8746 | AUC: 0.5843849206349206
📊 Confusion Matrix:
 [[359   0]
 [  0  27]]
```

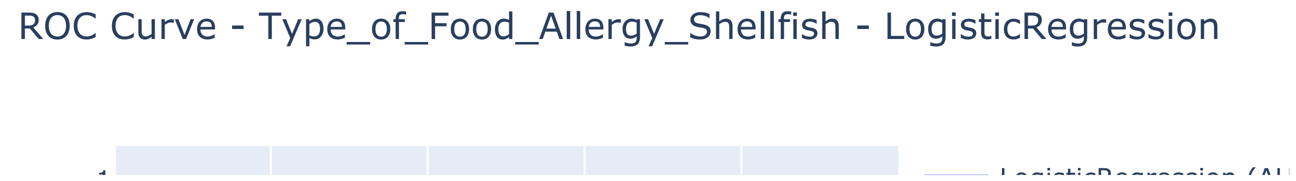
ROC Curve - Type_of_Food_Allergy_Shellfish - RandomForest

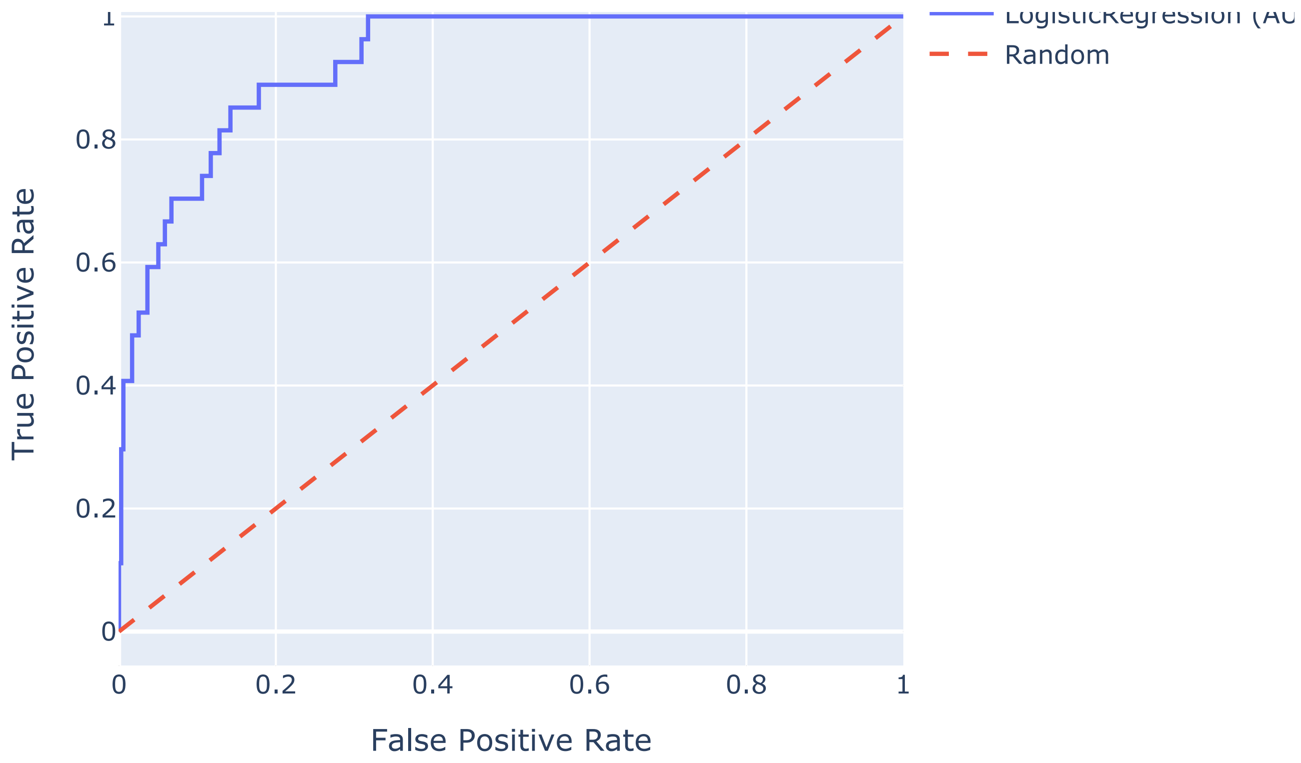


```
🔍 Target: Type_of_Food_Allergy_Shellfish | Model: XGBoost
📈 Accuracy: 0.9065
🎯 F1 (0): 0.9501 | F1 (1): 0.2333
📊 Precision: 0.9015 | AUC: 0.6359920634920635
📊 Confusion Matrix:
 [[359   0]
 [  0  27]]
```



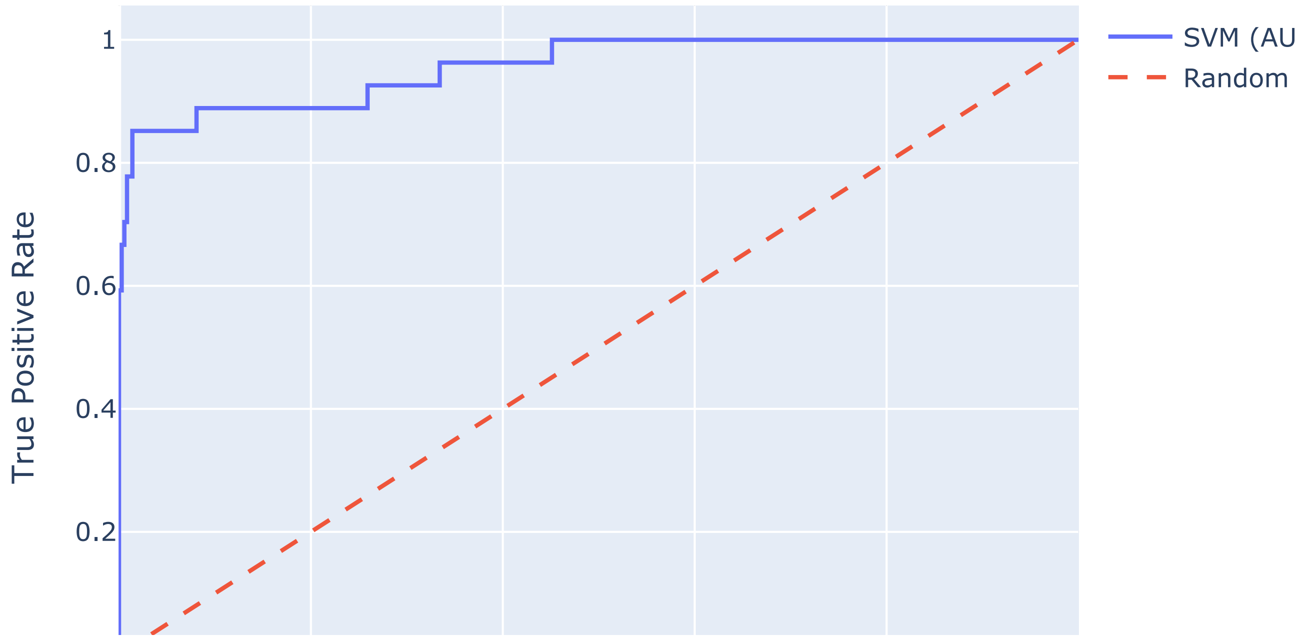
```
🔍 Target: Type_of_Food_Allergy_Shellfish | Model: LogisticRegression
📈 Accuracy: 0.8185
🎯 F1 (0): 0.8978 | F1 (1): 0.0867
📊 Precision: 0.8714 | AUC: 0.5184788359788359
📊 Confusion Matrix:
 [[359   0]
 [ 26   1]]
```

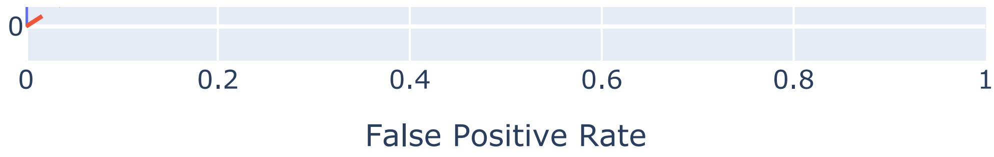




```
🔍 Target: Type_of_Food_Allergy_Shellfish | Model: SVM
📈 Accuracy: 0.7542
🎯 F1 (0): 0.8533 | F1 (1): 0.2070
📊 Precision: 0.8922 | AUC: 0.6400264550264552
📊 Confusion Matrix:
 [[359   0]
 [ 27   0]]
```

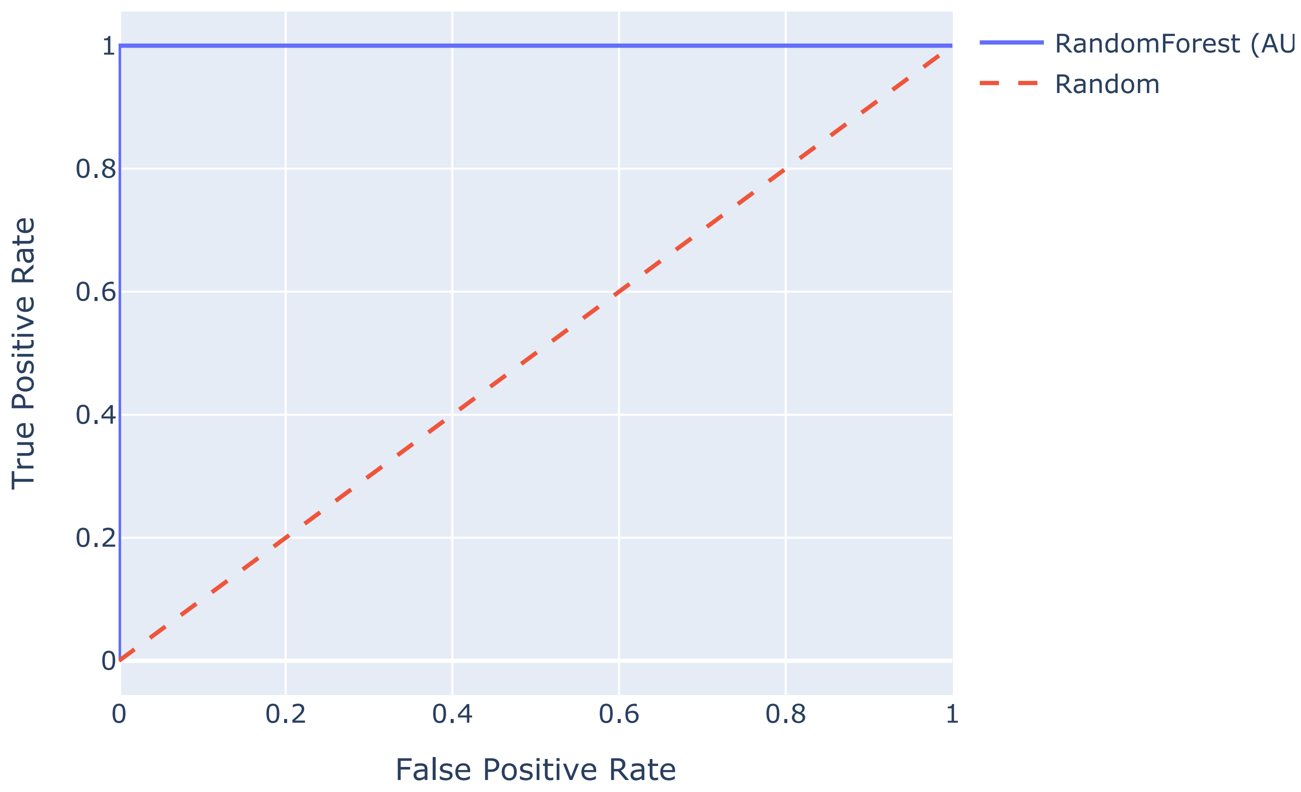
ROC Curve - Type_of_Food_Allergy_Shellfish - SVM





```
🔍 Target: Type_of_Food_Allergy_TPO | Model: RandomForest
📈 Accuracy: 0.9379
🎯 F1 (0): 0.9678 | F1 (1): 0.0000
📊 Precision: 0.8891 | AUC: 0.462743993993994
📊 Confusion Matrix:
 [[364   0]
 [  0  22]]
```

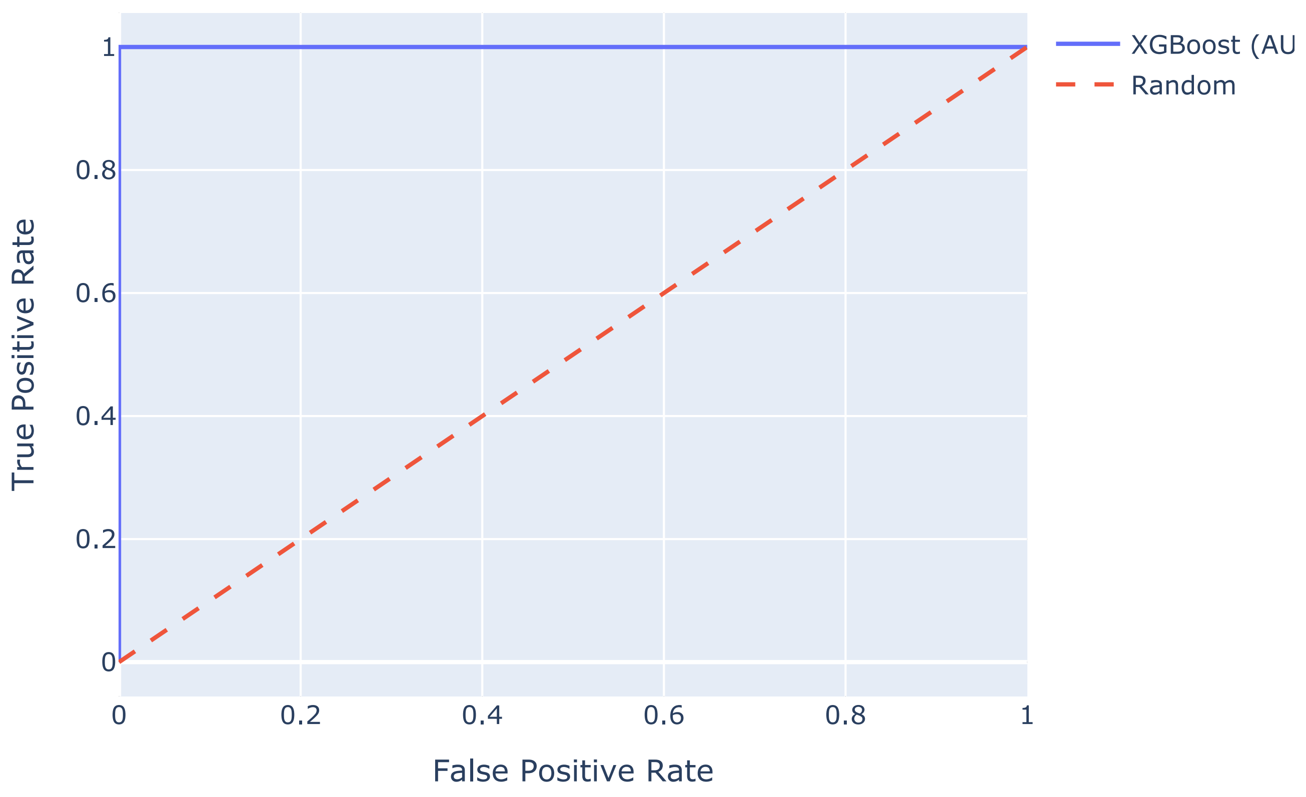
ROC Curve - Type_of_Food_Allergy_TPO - RandomForest



```
🔍 Target: Type_of_Food_Allergy_TPO | Model: XGBoost
📈 Accuracy: 0.9121
🎯 F1 (0): 0.9532 | F1 (1): 0.0000
📊 Precision: 0.8872 | AUC: 0.5221221221221221
📊 Confusion Matrix:
 [[364   0]
 [  0  22]]
```

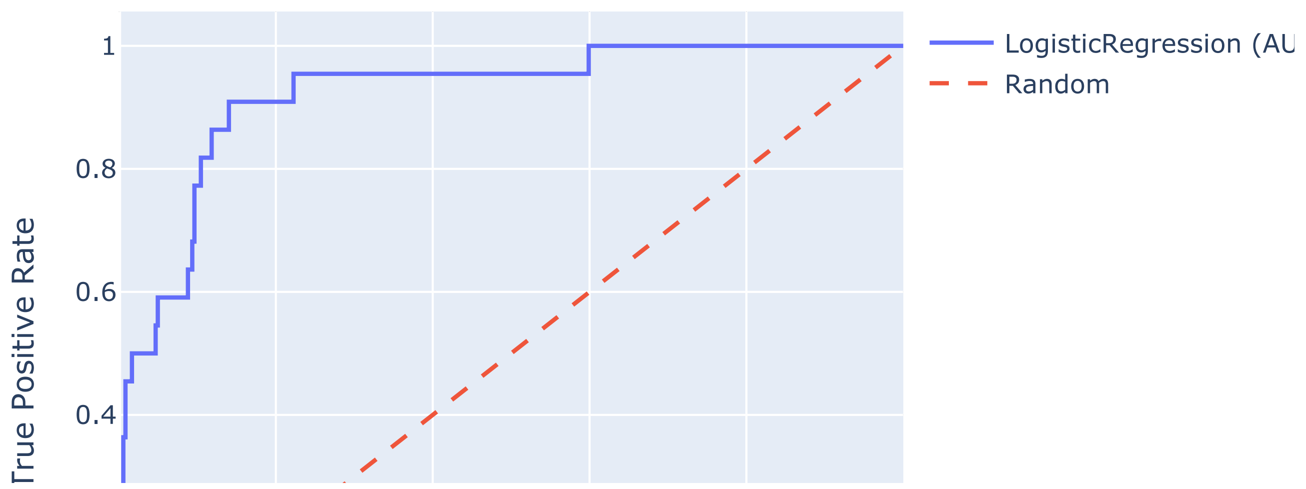
ROC Curve - Type_of_Food_Allergy_TPO - XGBoost

ROC Curve - Type_of_Food_Allergy_TPO - XGBoost



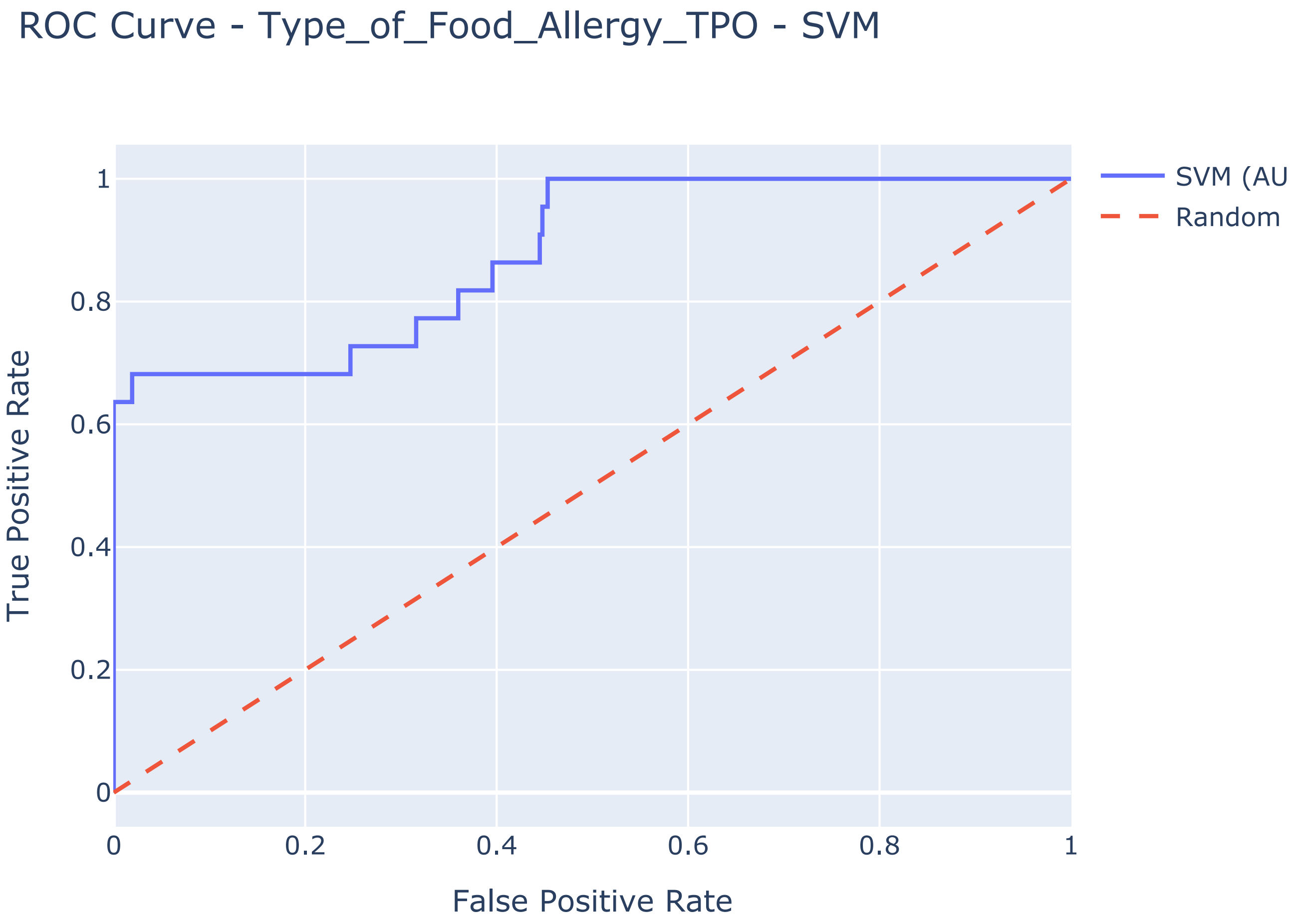
```
🔍 Target: Type_of_Food_Allergy_TPO | Model: LogisticRegression
📈 Accuracy: 0.8525
🎯 F1 (0): 0.9169 | F1 (1): 0.0958
📊 Precision: 0.8948 | AUC: 0.5379754754754755
📊 Confusion Matrix:
 [[363   1]
 [ 19   3]]
```

ROC Curve - Type_of_Food_Allergy_TPO - LogisticRegression



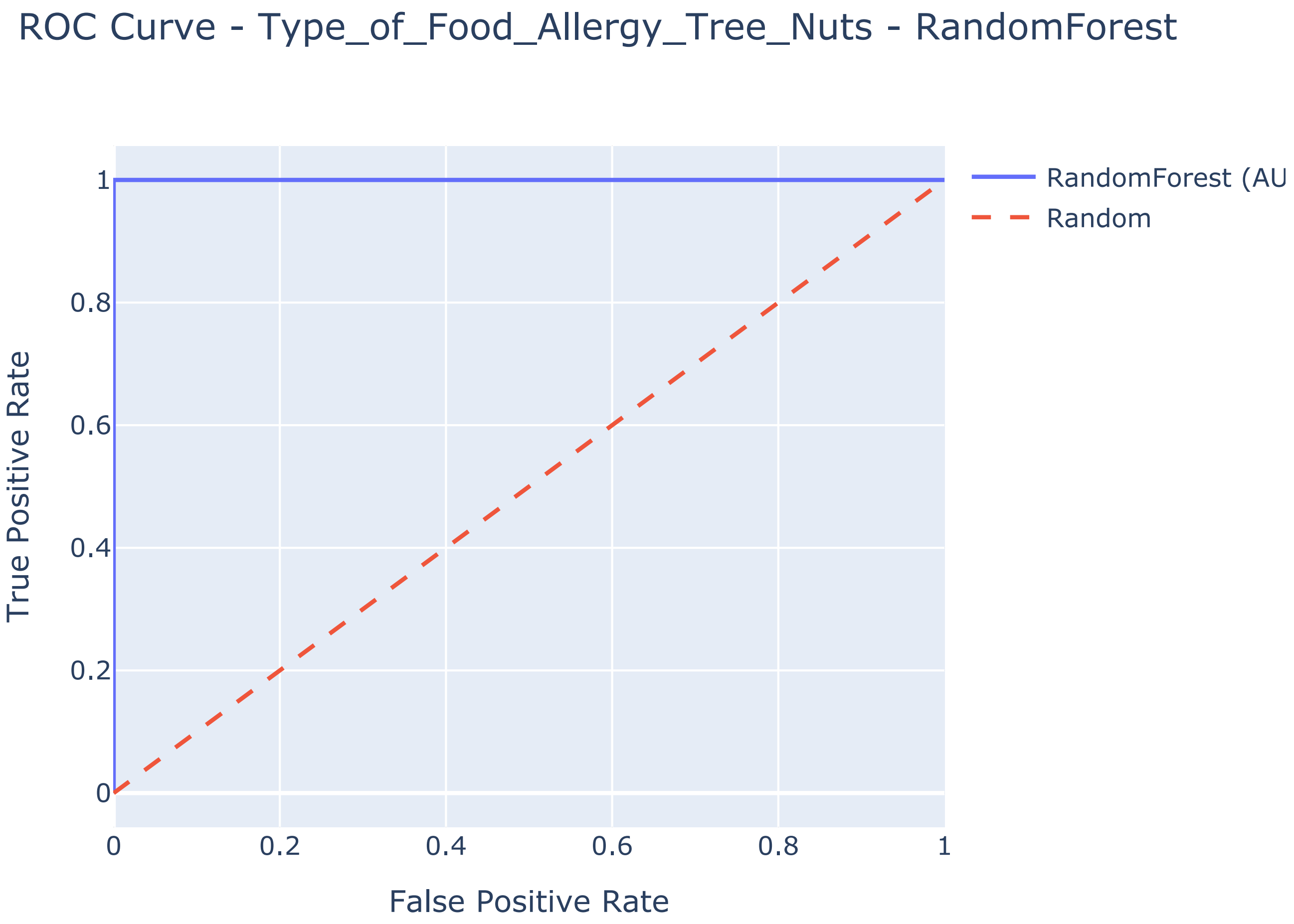
0.2
0
0
0.2
0.4
0.6
0.8
1
False Positive Rate

```
🔍 Target: Type_of_Food_Allergy_TPO | Model: SVM
📈 Accuracy: 0.7875
🎯 F1 (0): 0.8780 | F1 (1): 0.1406
📊 Precision: 0.9036 | AUC: 0.6246746746746747
📊 Confusion Matrix:
 [[364   0]
 [ 22   0]]
```

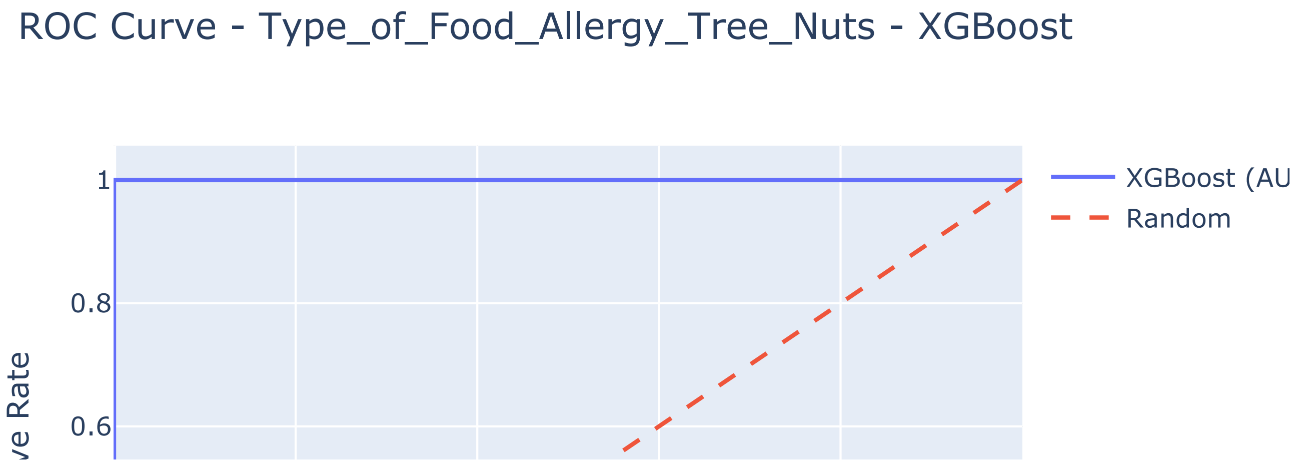


```
🔍 Target: Type_of_Food_Allergy_Tree_Nuts | Model: RandomForest
📈 Accuracy: 0.7252
🎯 F1 (0): 0.8305 | F1 (1): 0.2604
📊 Precision: 0.6982 | AUC: 0.6592209450830141
📊 Confusion Matrix:
```

```
[[292   0]
 [  0  94]]
```

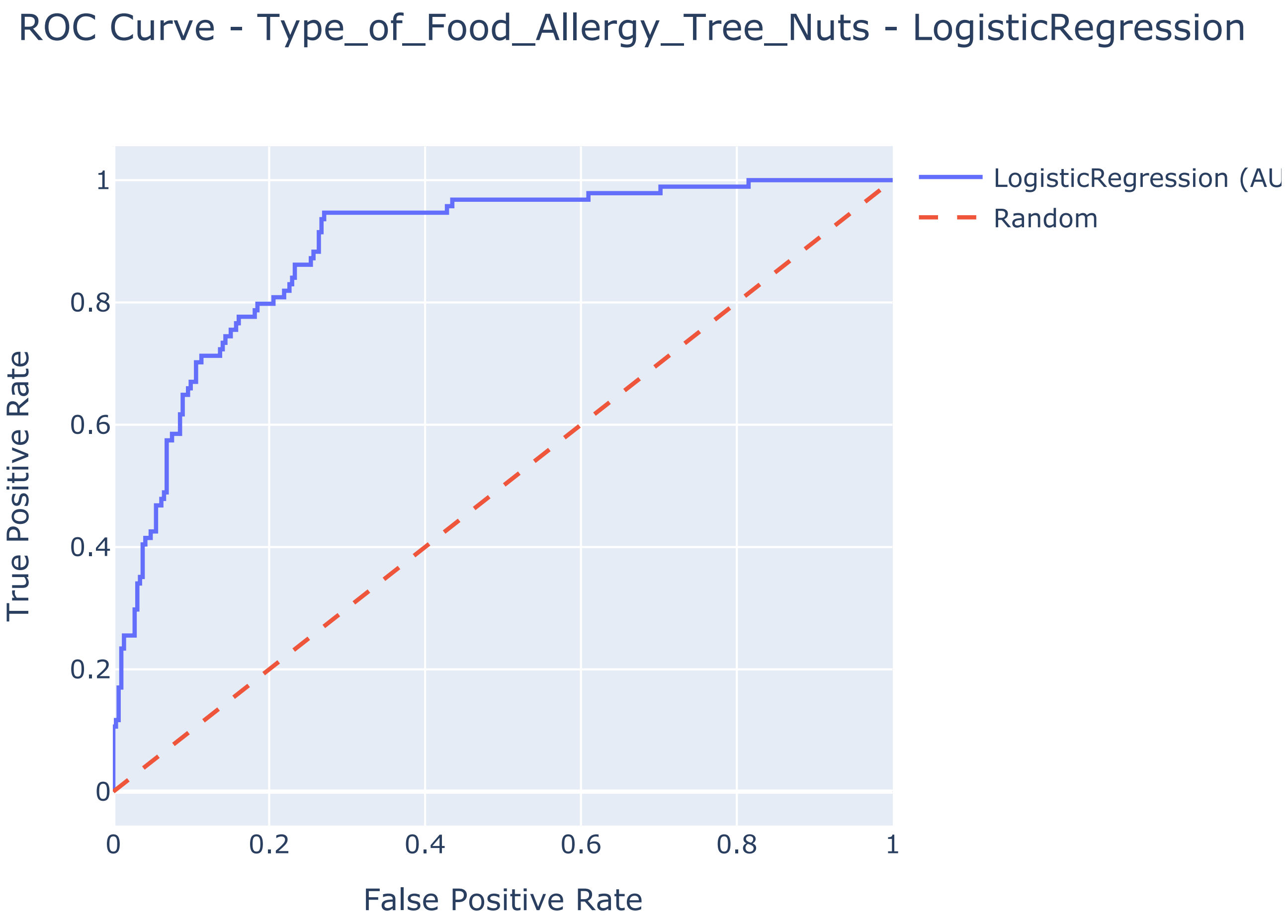


```
🔍 Target: Type_of_Food_Allergy_Tree_Nuts | Model: XGBoost
📈 Accuracy: 0.7410
🎯 F1 (0): 0.8304 | F1 (1): 0.4378
📊 Precision: 0.7404 | AUC: 0.7172286079182632
📊 Confusion Matrix:
[[292   0]
 [  0  94]]
```

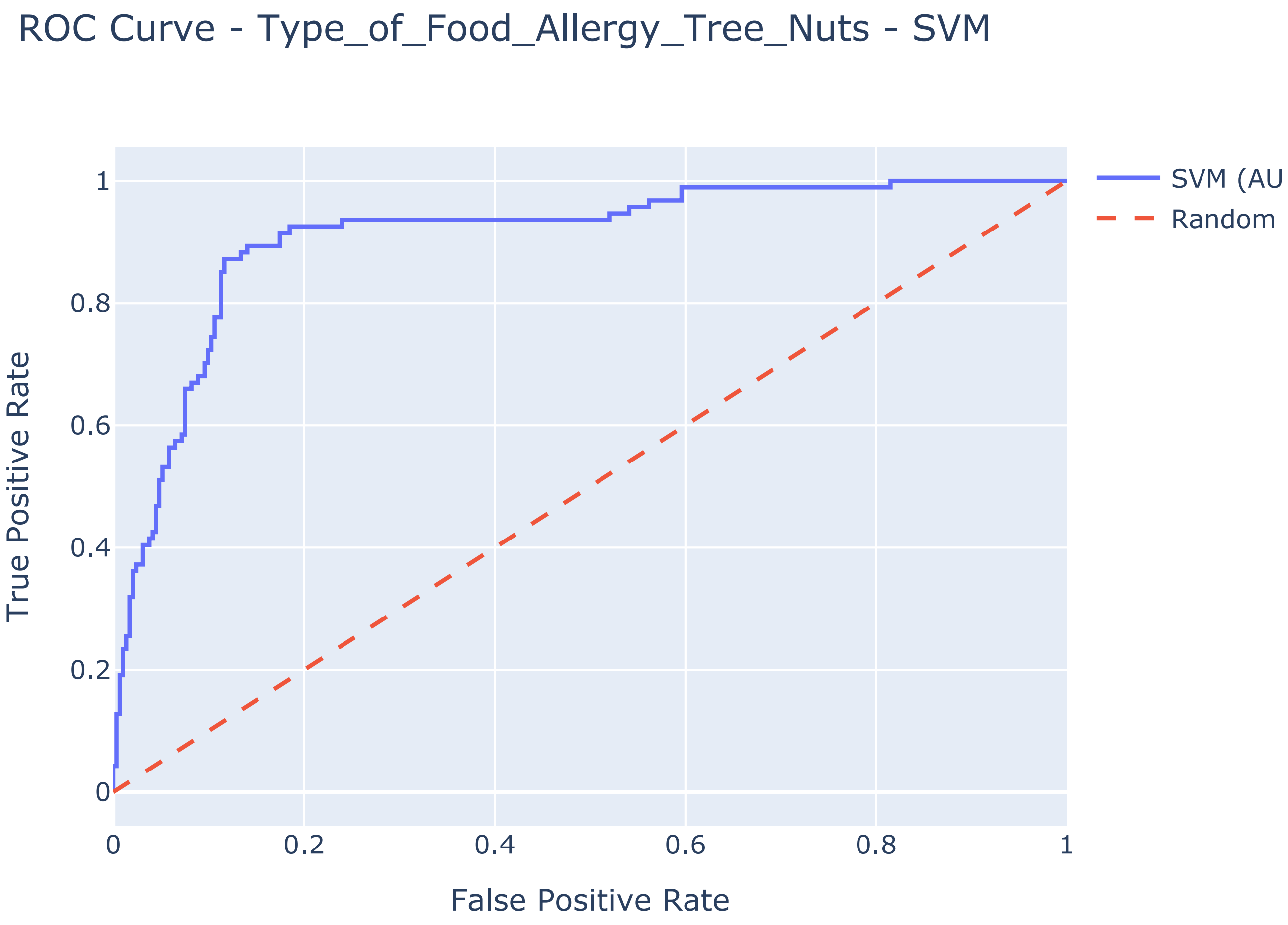


True Positi
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
False Positive Rate

```
🔍 Target: Type_of_Food_Allergy_Tree_Nuts | Model: LogisticRegression
📈 Accuracy: 0.6922
🎯 F1 (0): 0.7935 | F1 (1): 0.3754
📊 Precision: 0.7029 | AUC: 0.6821072796934866
📊 Confusion Matrix:
 [[279  13]
 [ 55  39]]
```



```
🔍 Target: Type_of_Food_Allergy_Tree_Nuts | Model: SVM
🔍 Target: Type_of_Food_Allergy_Tree_Nuts | Model: SVM
📈 Accuracy: 0.6213
🎯 F1 (0): 0.7105 | F1 (1): 0.4414
📊 Precision: 0.7190 | AUC: 0.6295913154533844
📊 Confusion Matrix:
 [[292   0]
 [ 93   1]]
```



```
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedKFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.metrics import (
    f1_score, accuracy_score, recall_score,
    precision_score, confusion_matrix, roc_auc_score, roc_curve
)
from imblearn.over_sampling import SMOTE
import plotly.graph_objects as go
```

```
ALEX_venom = ALEX[ALEX["Venom_Allergy"] == 1]
targets = ["Type_of_Venom_Allergy_ATCD_Venom",
    "Type_of_Venom_Allergy_IGE_Venom"]

models = {
    "RandomForest": RandomForestClassifier(random_state=42),
    "XGBoost": XGBClassifier(random_state=42, eval_metric="logloss", use_label_
    "LogisticRegression": LogisticRegression(max_iter=1000, random_state=42),
    "SVM": SVC(probability=True, random_state=42)
}

X=ALEX_venom.copy()
X.drop(target_1, axis=1, inplace=True)
X.drop(extra_columns, axis=1, inplace=True)
X.drop(extra, axis=1, inplace=True)
X.drop(inconnu, axis=1, inplace=True)
X = X.iloc[:, 1:]
results_venom = []

kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

for target in targets:
    y = ALEX_venom[target]

    for model_name, base_model in models.items():
        f1_class0_scores, f1_class1_scores = [], []
        precision_scores, acc_scores, recall_scores, auc_scores = [], [], [], [

        for train_idx, test_idx in kfold.split(X, y):
            X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
            y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]

            smote = SMOTE(random_state=42)
            X_train_res, y_train_res = smote.fit_resample(X_train, y_train)

            base_model.fit(X_train_res, y_train_res)
            y_pred = base_model.predict(X_test)

            acc_scores.append(accuracy_score(y_test, y_pred))
            recall_scores.append(recall_score(y_test, y_pred, zero_division=0))
            precision_scores.append(precision_score(y_test, y_pred, average='we
            f1_class0_scores.append(f1_score(y_test, y_pred, pos_label=0, zero_
            f1_class1_scores.append(f1_score(y_test, y_pred, pos_label=1, zero_

            if hasattr(base_model, "predict_proba"):
                y_proba = base_model.predict_proba(X_test)[:, 1]
                auc_scores.append(roc_auc_score(y_test, y_proba))
        base_model.fit(X, y)
        y_pred_full = base_model.predict(X)
        y_proba_full = base_model.predict_proba(X)[:, 1] if hasattr(base_model,
        matrix = confusion_matrix(y, y_pred_full)

        print(f"\n🔍 Target: {target} | Model: {model_name}")
        print(f"📈 Accuracy: {np.mean(acc_scores):.4f}")
        print(f"🎯 F1 (0): {np.mean(f1_class0_scores):.4f} | F1 (1): {np.mean(
        print(f"📊 Precision: {np.mean(precision_scores):.4f} | AUC: {np.mean(a
        print("📊 Confusion Matrix:\n", matrix)

        if y_proba_full is not None:
            fpr, tpr, _ = roc_curve(y, y_proba_full)
            fig = go.Figure()
            fig.add_trace(go.Scatter(x=fpr, y=tpr, mode='lines', name=f"{model_
            fig.add_trace(go.Scatter(x=[0, 1], y=[0, 1], mode='lines', name='Ra
            fig.update_layout(
                title=f"ROC Curve - {target} - {model_name}",
                xaxis_title="False Positive Rate",
                yaxis_title="True Positive Rate",
                width=700, height=500
            )
            fig.show()

        results_venom.append({
            "Target": target,
            "Model": model_name,
            "F1_Class_0": np.mean(f1_class0_scores),
            "F1_Class_1": np.mean(f1_class1_scores),
            "Precision": np.mean(precision_scores),
            "Accuracy": np.mean(acc_scores),
            "Recall": np.mean(recall_scores),
            "AUC_ROC": np.mean(auc_scores) if auc_scores else np.nan
        })

pd.DataFrame(results_venom).to_csv("results_ALEX_venom.csv", index=False)
```

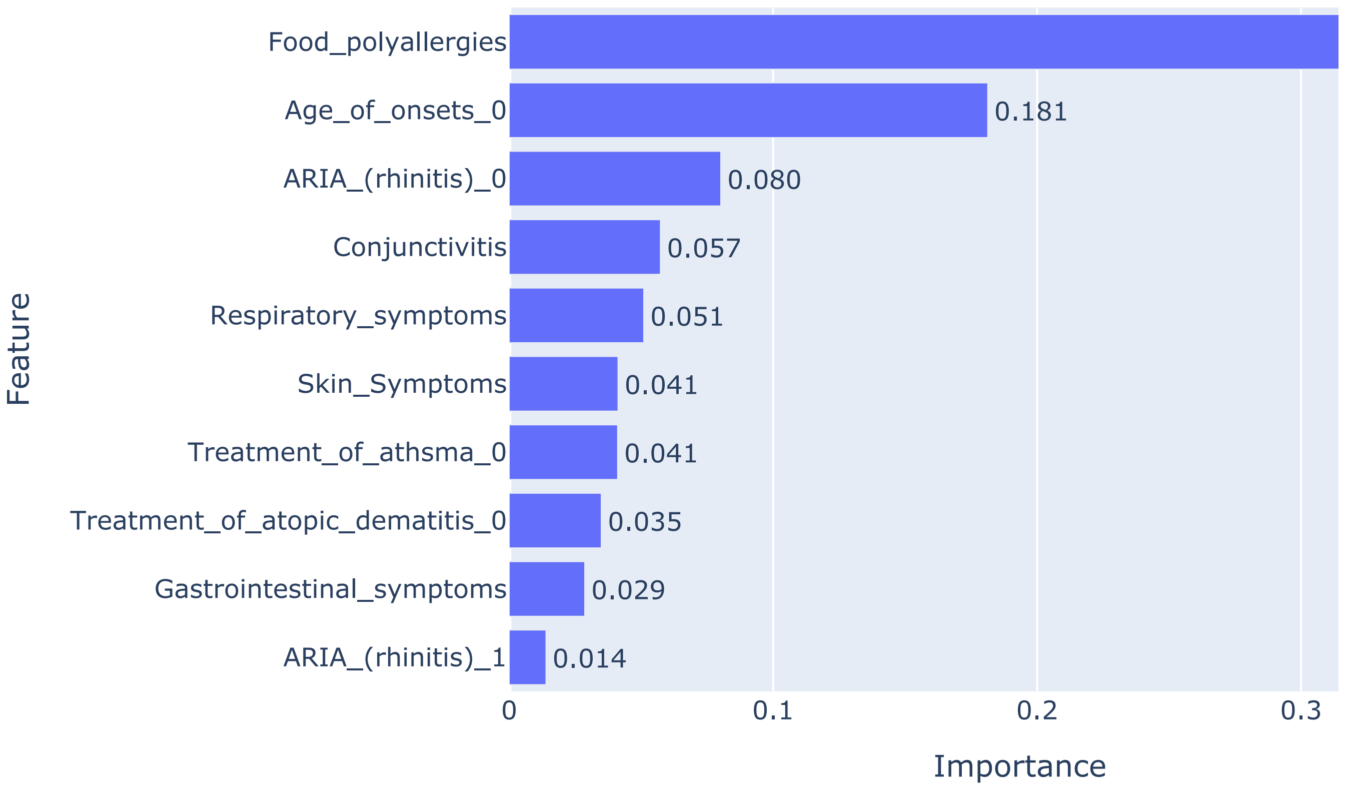
Ne lancer pas cette partie, c pour la recherche des hyperparametres

[ ] ↳ 1 cell hidden

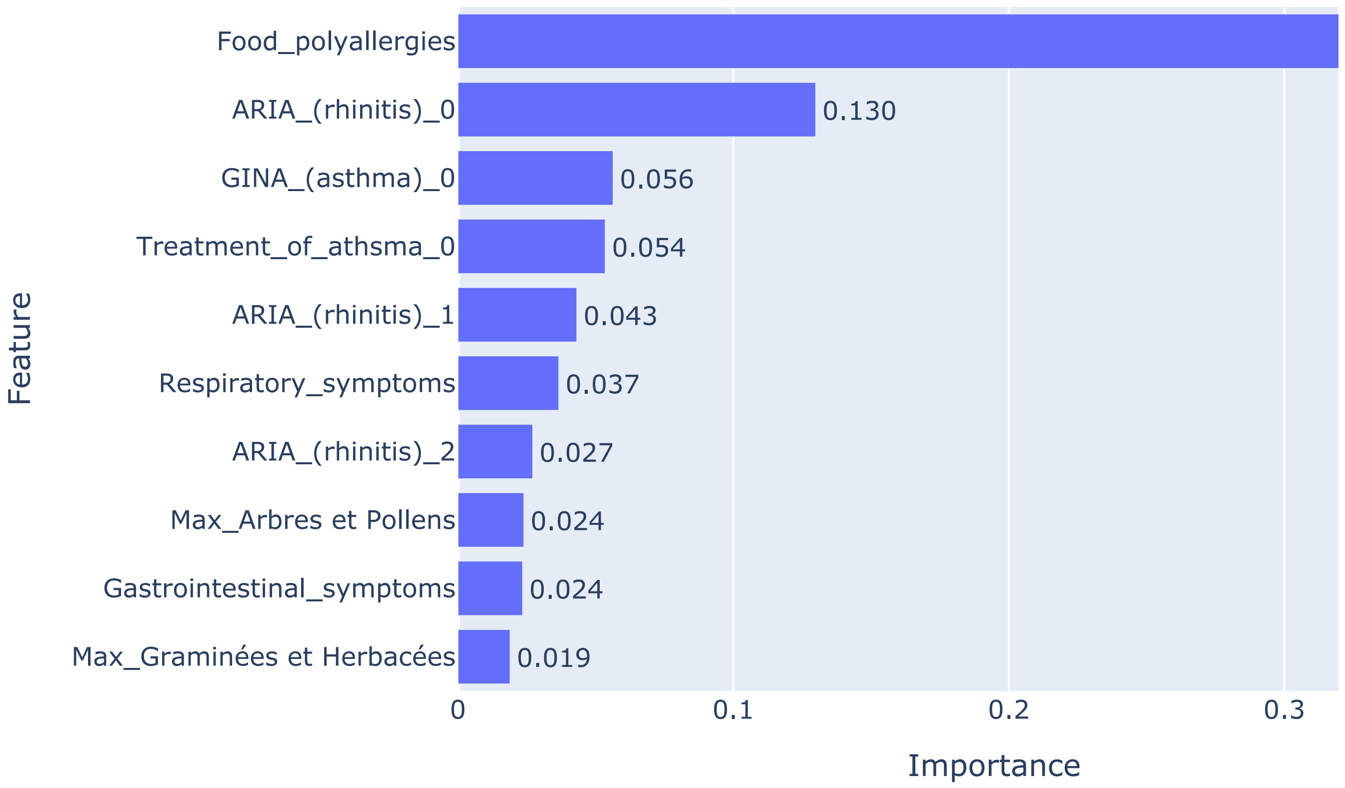
TOP Features

```
import pandas as pd
import numpy as np
from xgboost import XGBClassifier
import plotly.graph_objects as go

targets = [
    "Allergy_Present", "Respiratory_Allergy", "Food_Allergy", "Venom_Allergy",
    "Severe_Allergy", "Type_of_Food_Allergy_Other", "Type_of_Respiratory_Allergy
    "Type_of_Respiratory_Allergy_IGE_Pollen_Tree", "Type_of_Respiratory_Allergy_
    "Type_of_Respiratory_Allergy_IGE_Mite_Cockroach", "Type_of_Respiratory_Aller
    "Type_of_Respiratory_Allergy_ARIA", "Type_of_Respiratory_Allergy_CONJ",
    "Type_of_Respiratory_Allergy_IGE_Pollen_Gram", "Type_of_Respiratory_Allergy_
    "Type_of_Food_Allergy_Aromatics", "Type_of_Food_Allergy_Cereals_&_Seeds",
    "Type_of_Food_Allergy_Egg", "Type_of_Food_Allergy_Fish", "Type_of_Food_Aller
    "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_
    "Type_of_Venom_Allergy_ATCD_Venom", "Type_of_Venom_Allergy_IGE_Venom"
]

inconnu = ["Treatment_of_athsma_9", "Treatment_of_rhinitis_9", "General_cofactor
           "Age_of_onsets_9", "ARIA_(rhinitis)_9", "GINA_(asthma)_9", "Treatment

X = ALEX.copy()
X.drop(target_1, axis=1, inplace=True)
X.drop(extra_columns, axis=1, inplace=True)
X.drop(extra, axis=1, inplace=True)
X.drop(inconnu, axis=1, inplace=True)
X = X.iloc[:, 1:]

def plot_top_features(model, X_sub, y_sub, target):
    if len(np.unique(y_sub)) < 2:
        print(f"⚠️ Target '{target}' contient une seule classe ({np.unique(y_sub
        return

    model.fit(X_sub, y_sub)
    importances = model.feature_importances_
    top_indices = np.argsort(importances)[::-1][:10]
    features = X_sub.columns[top_indices]
    scores = importances[top_indices]

    fig = go.Figure(go.Bar(
        x=scores[::-1],
        y=features[::-1]
        y=features[::-1],
        orientation='h',
        text=[f"{s:.3f}" for s in scores[::-1]],
        textposition='outside'
    ))
    fig.update_layout(
        title=f"Top 10 Features pour la cible '{target}' (XGBoost)",
        xaxis_title="Importance",
        yaxis_title="Feature",
        width=800, height=500
    )
    fig.show()

for target in targets:
    X_sub = X.copy()
    y_sub = ALEX[target]
    model = XGBClassifier(random_state=42, eval_metric="logloss", use_label_enco
    plot_top_features(model, X_sub, y_sub, target)
```

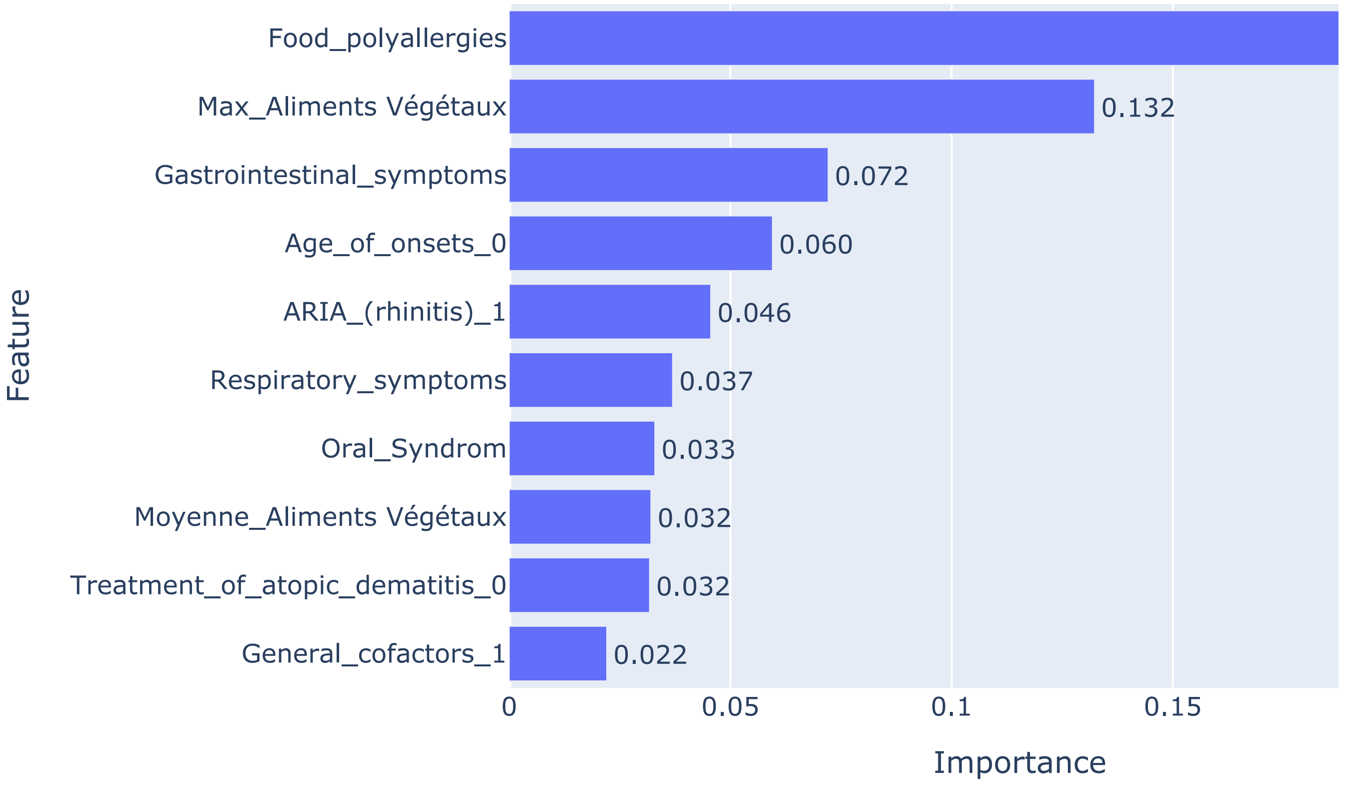
Top 10 Features pour la cible 'Allergy_Present' (XGBoost)

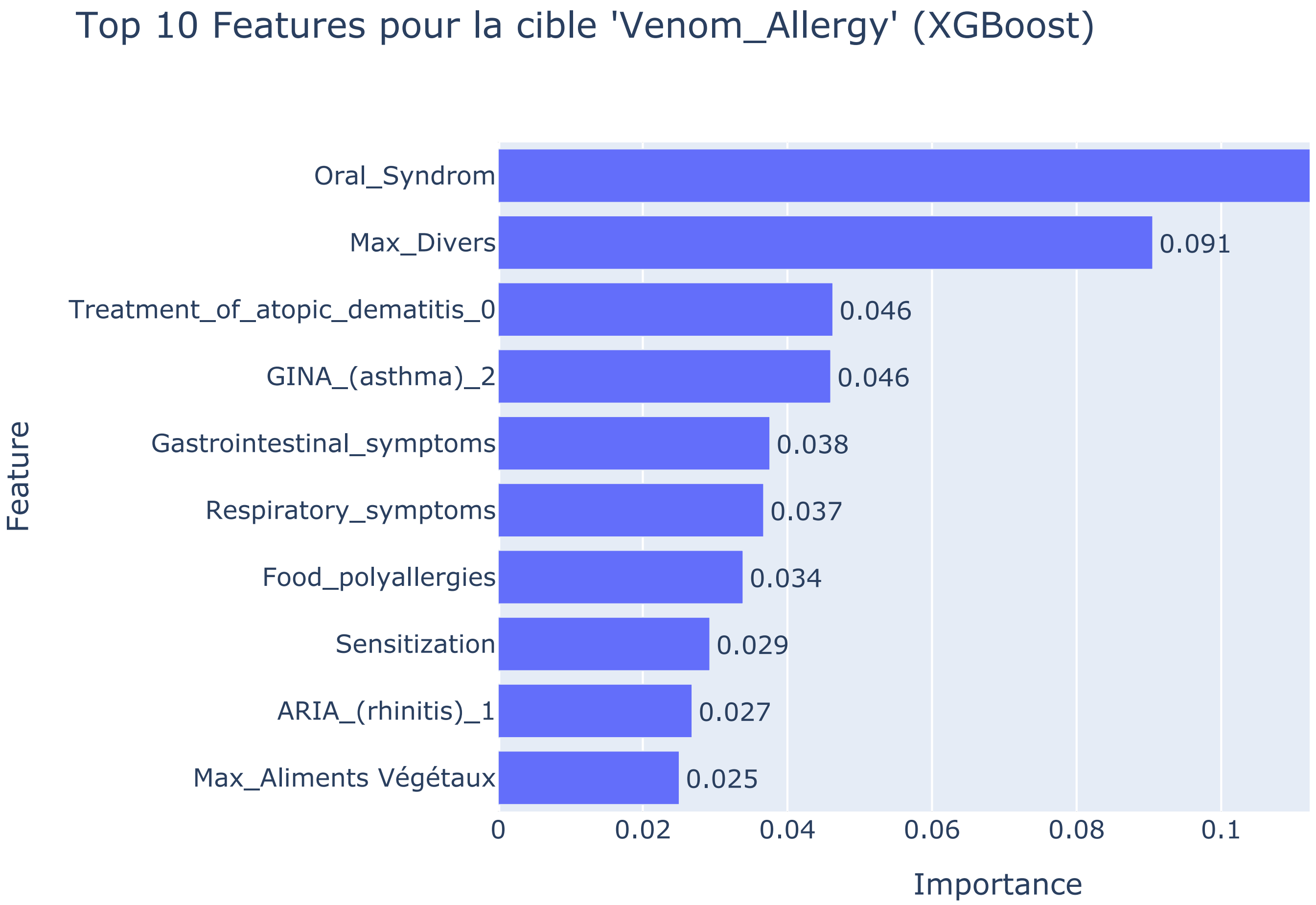


Top 10 Features pour la cible 'Respiratory_Allergy' (XGBoost)

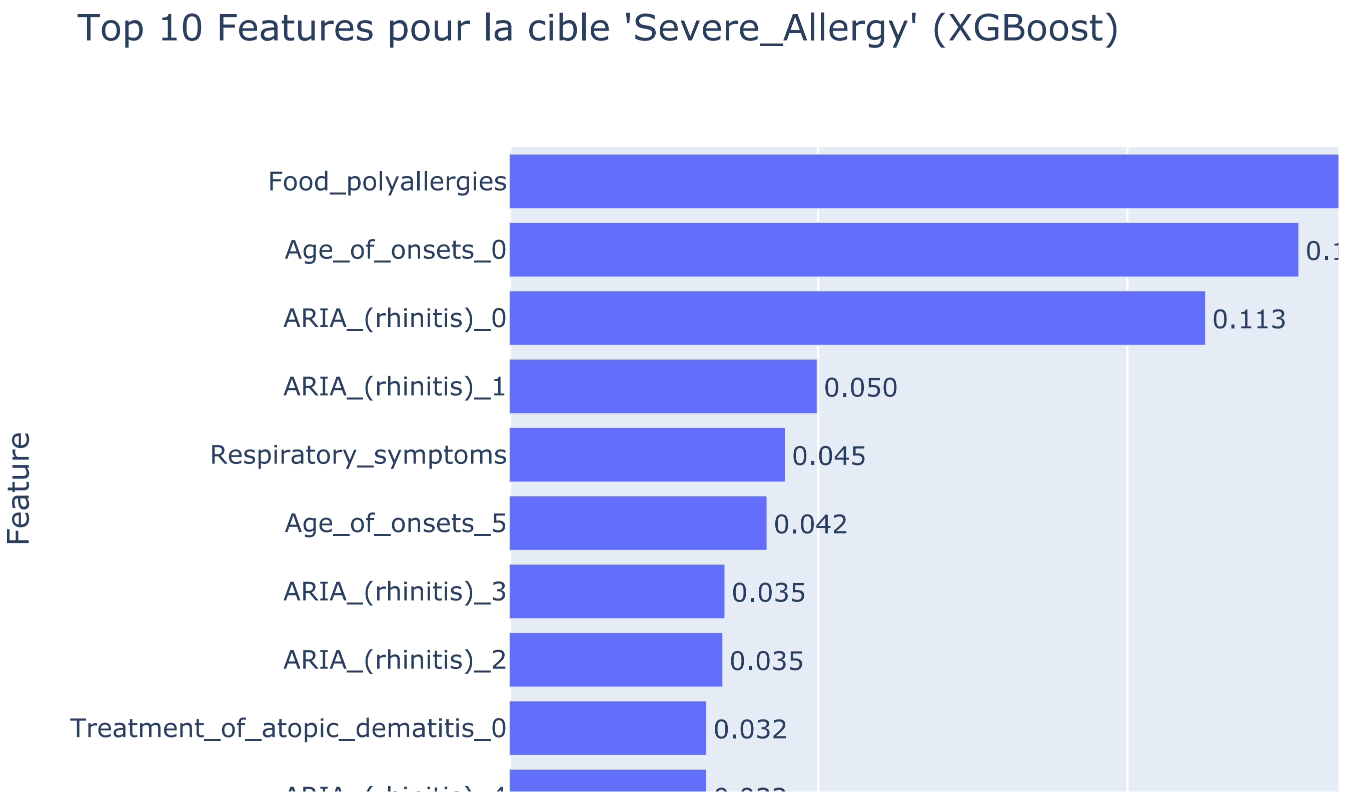


Top 10 Features pour la cible 'Food_Allergy' (XGBoost)

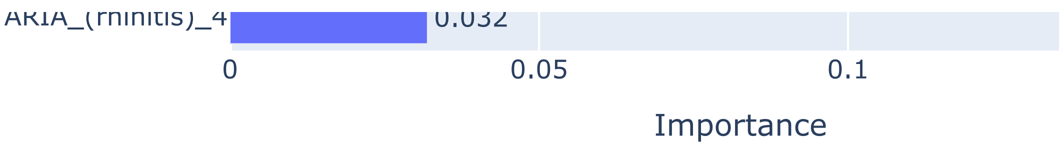




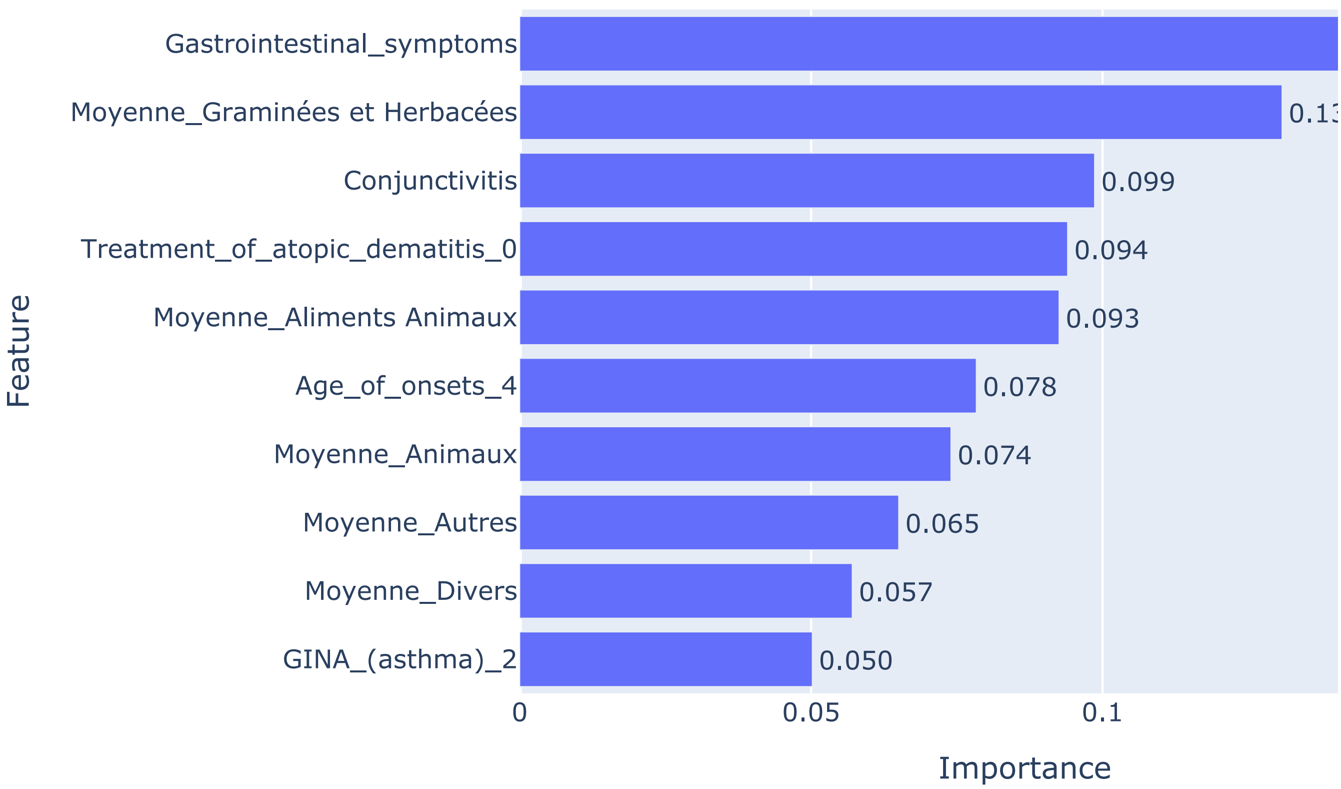
Top 10 Features pour la cible 'Venom_Allergy' (XGBoost)



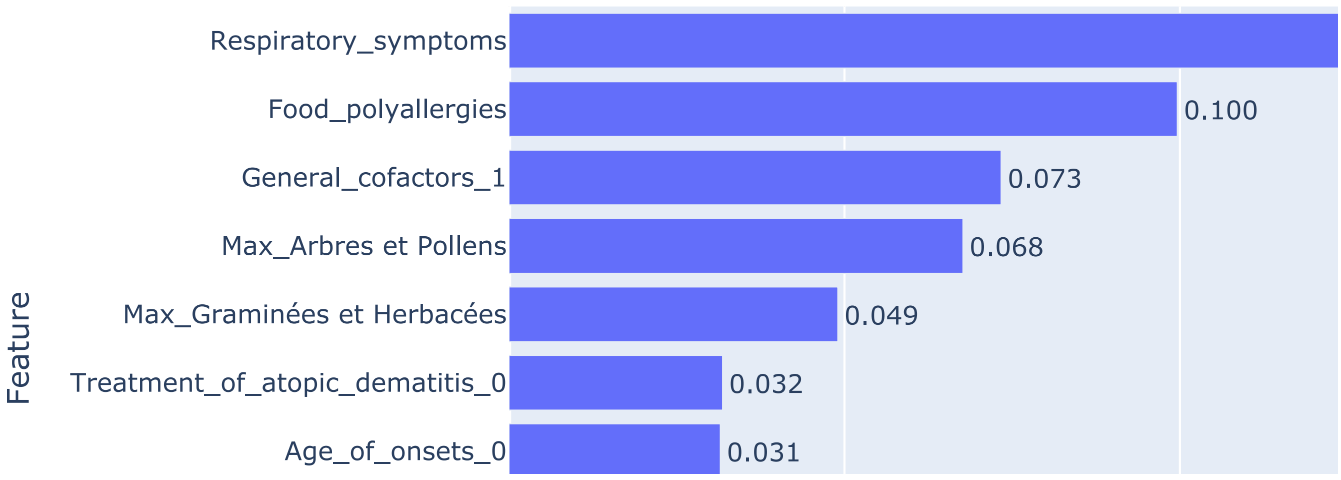
Top 10 Features pour la cible 'Severe_Allergy' (XGBoost)

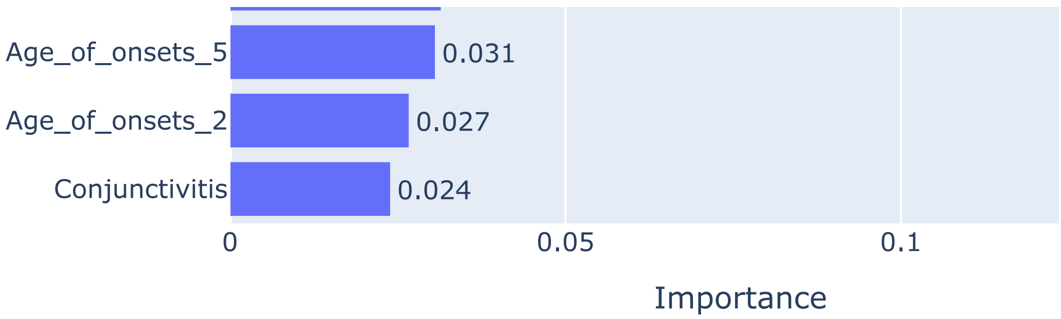


Top 10 Features pour la cible 'Type_of_Food_Allergy_Other' (XGBoos

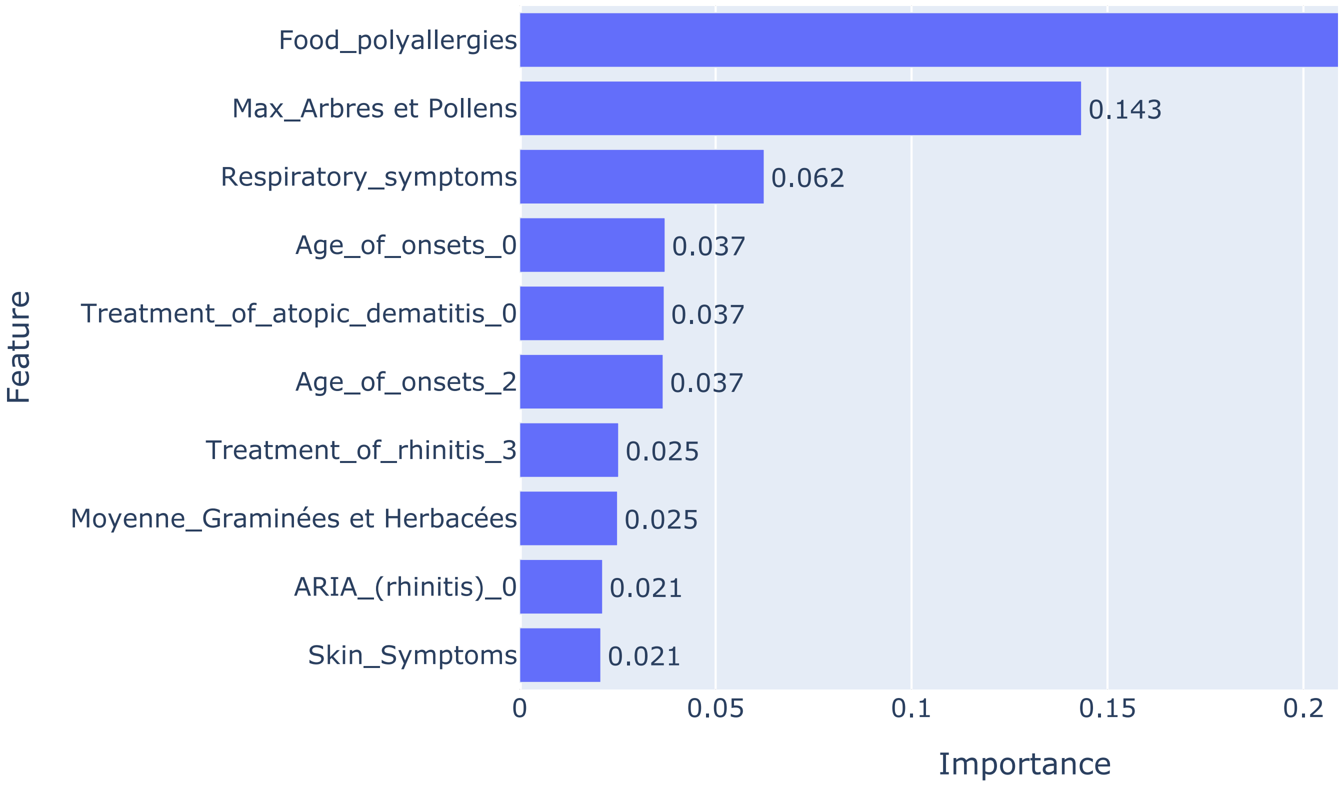


Top 10 Features pour la cible 'Type_of_Respiratory_Allergy_IGE_Poll

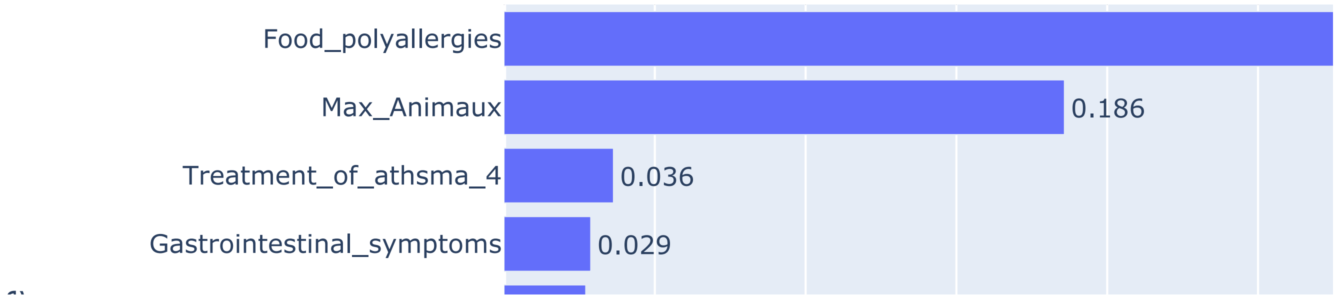


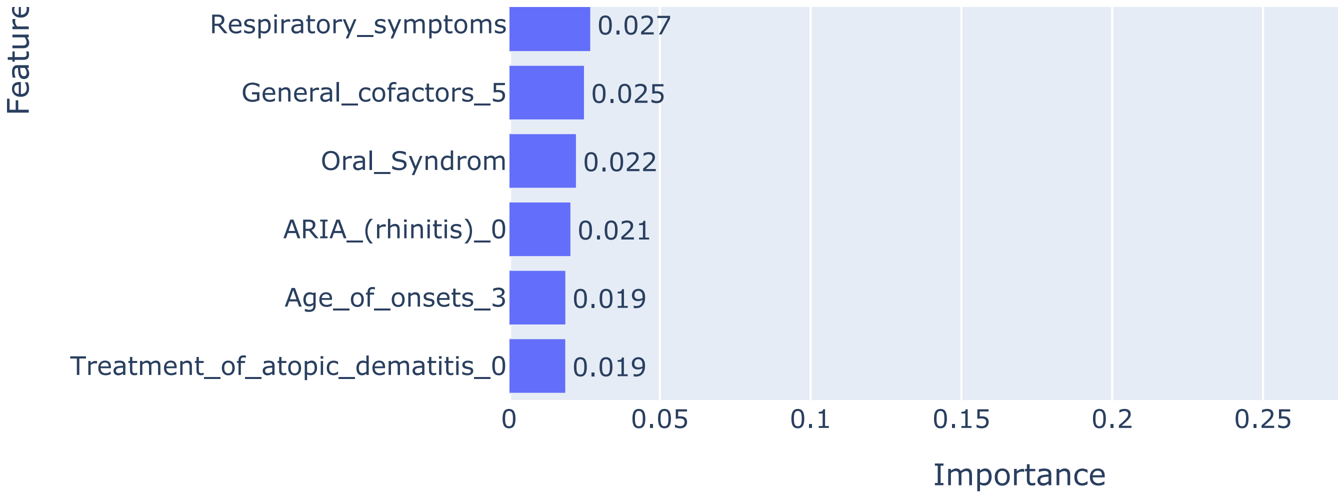


Top 10 Features pour la cible 'Type_of_Respiratory_Allergy_IGE_Poll

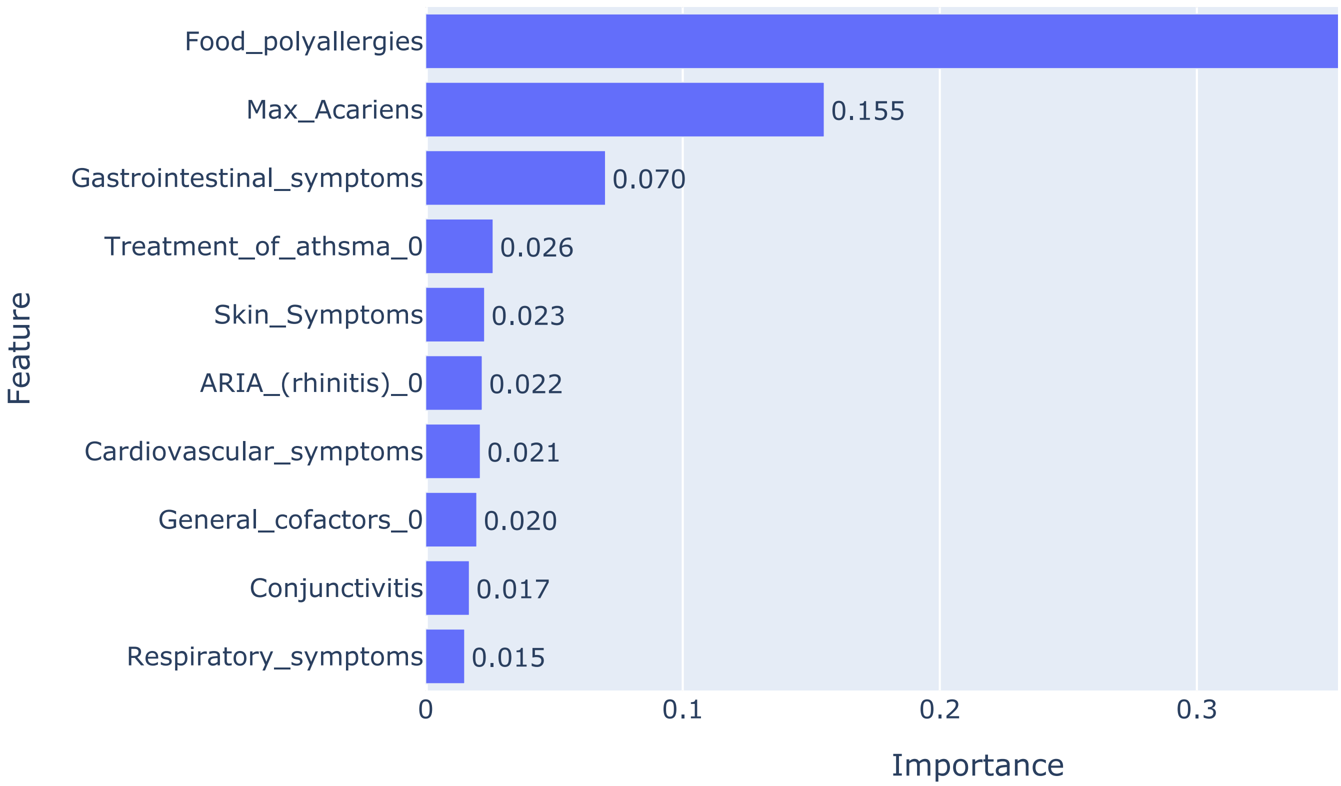


Top 10 Features pour la cible 'Type_of_Respiratory_Allergy_IGE_Dar



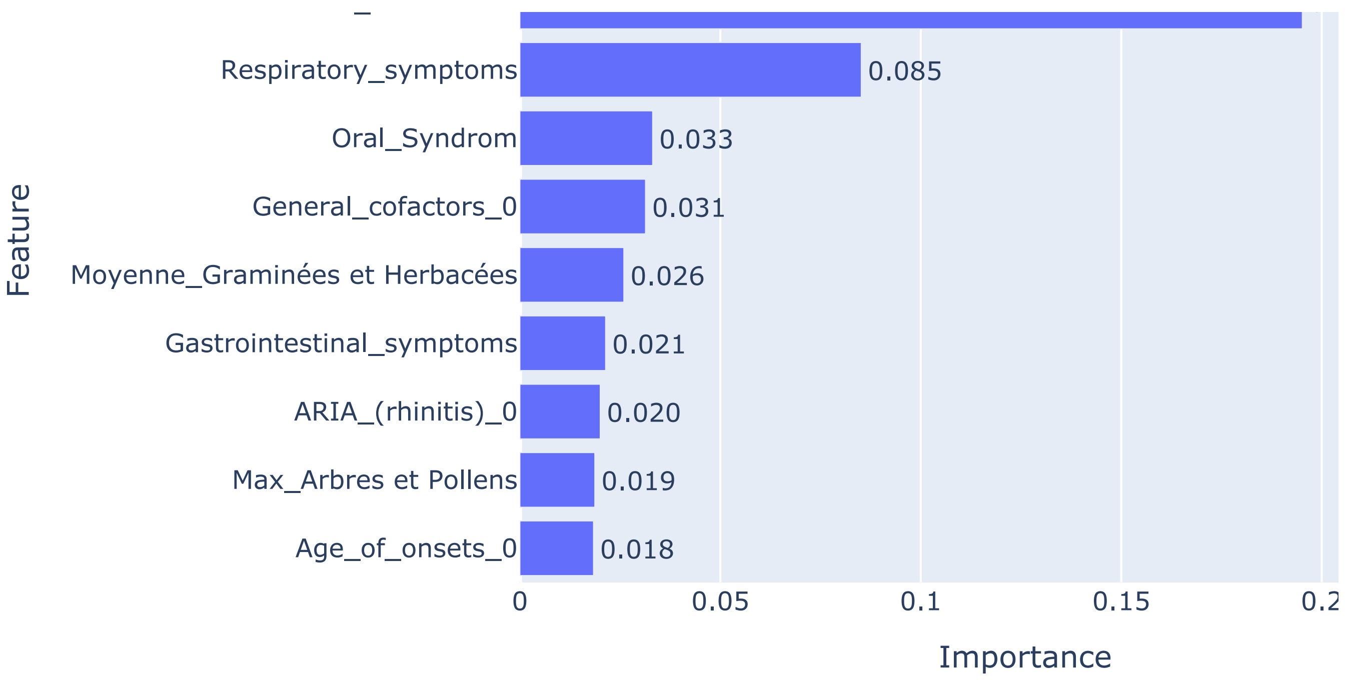


Top 10 Features pour la cible 'Type_of_Respiratory_Allergy_IGE_Mite

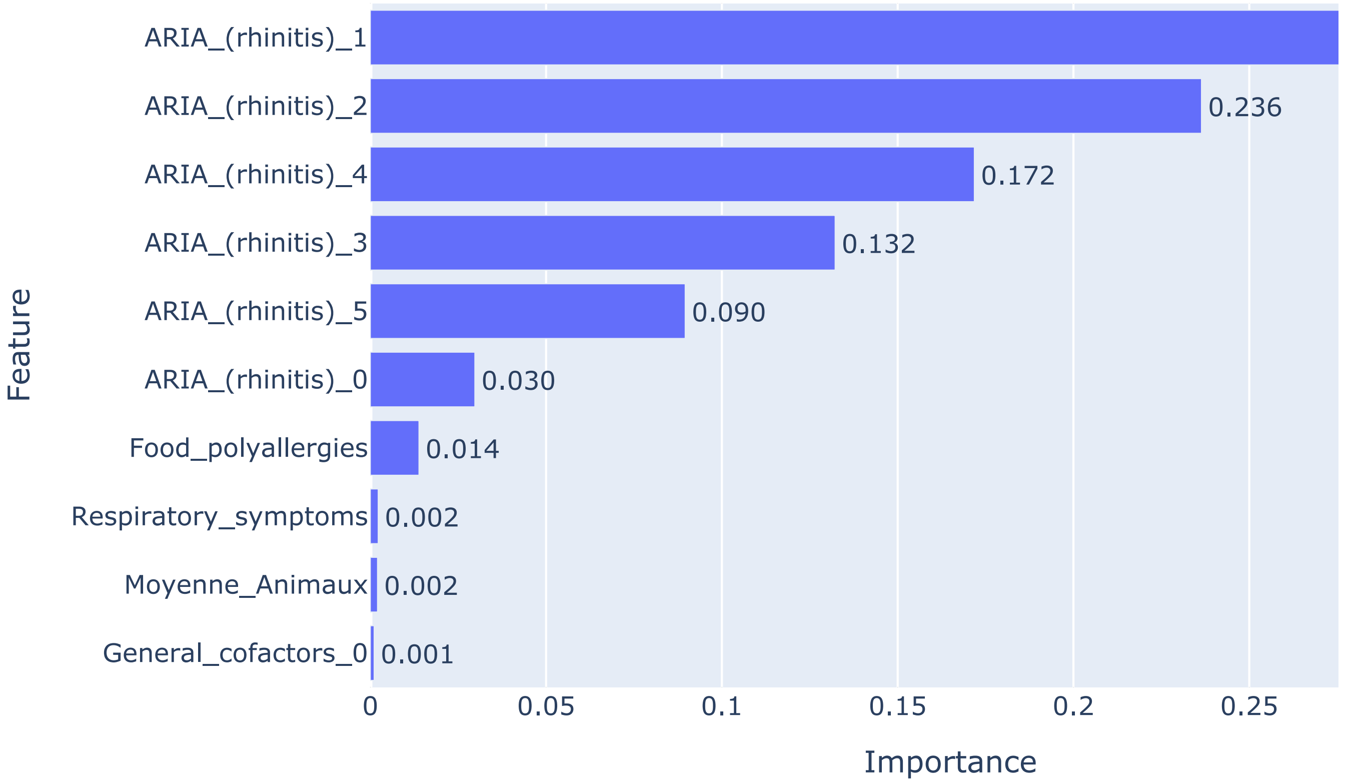


Top 10 Features pour la cible 'Type_of_Respiratory_Allergy_IGE_Mol

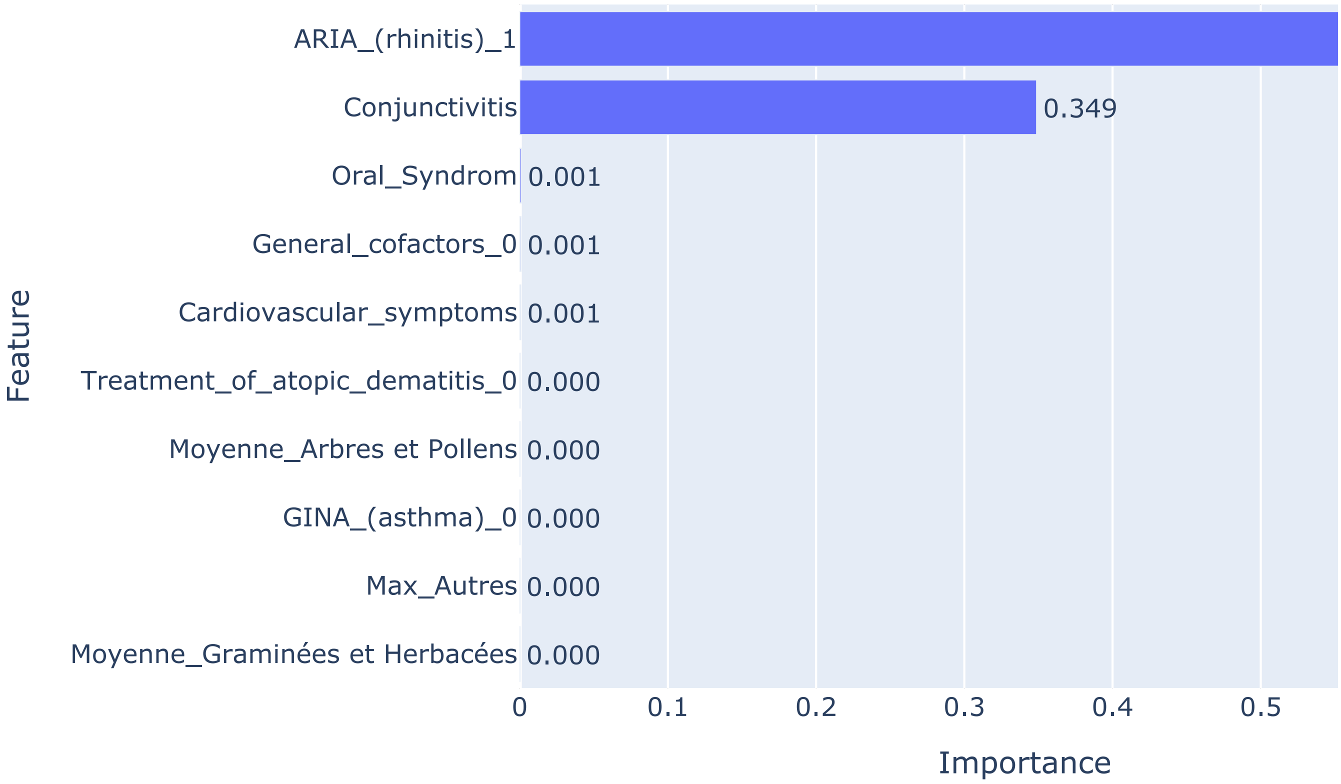
Food_polyallergies
Max_Moisissures 0.



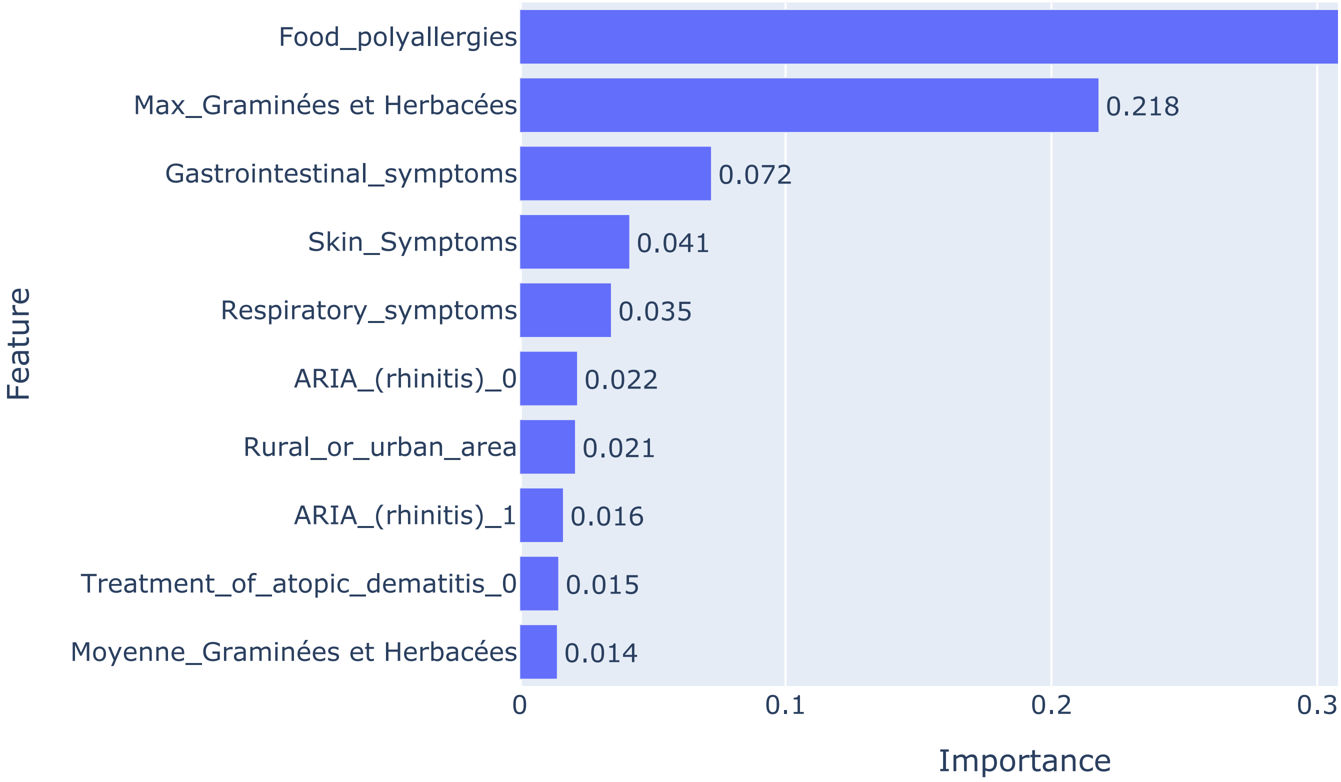
Top 10 Features pour la cible 'Type_of_Respiratory_Allergy_ARIA' (X



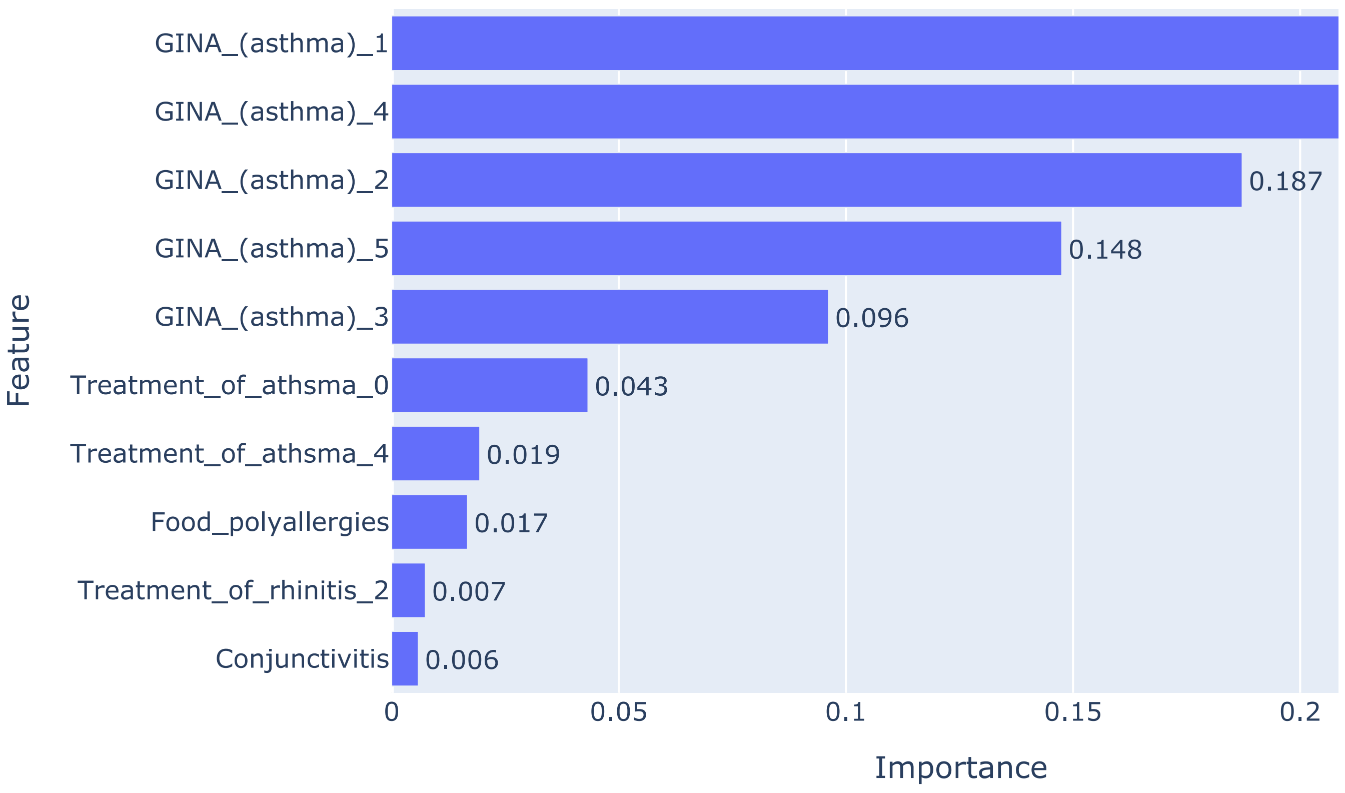
Top 10 Features pour la cible 'Type_of_Respiratory_Allergy_CONJ' (X



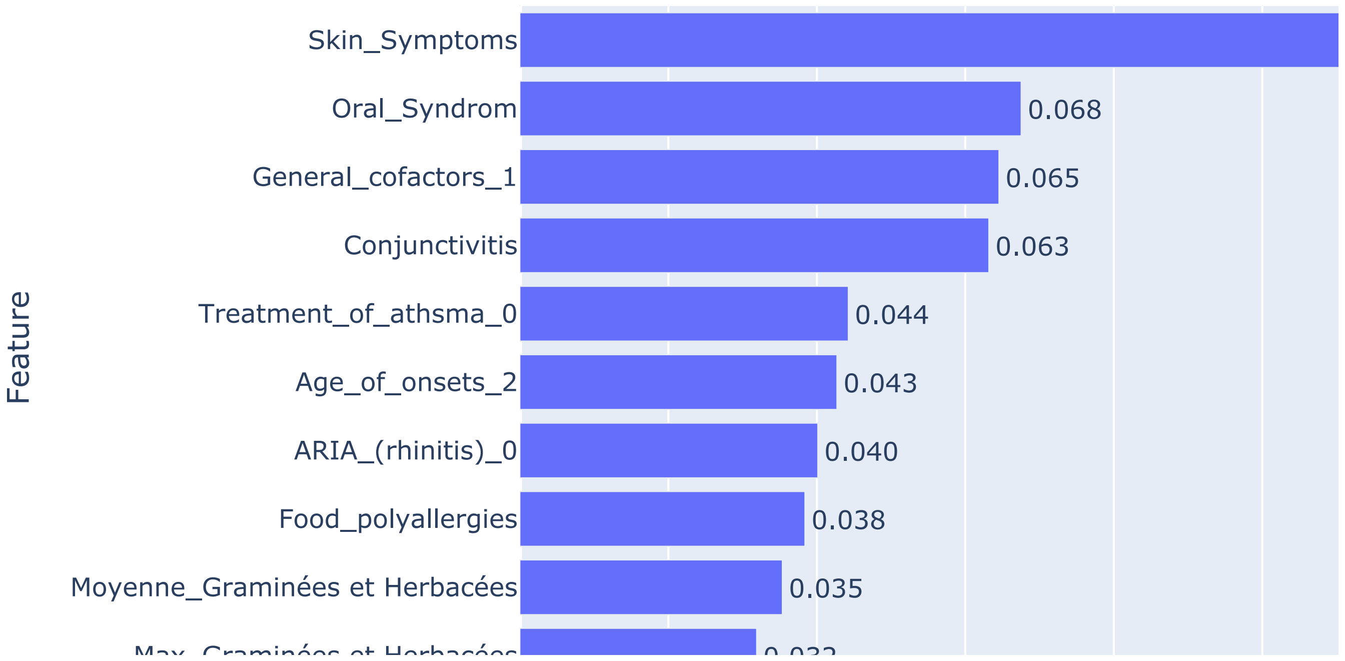
Top 10 Features pour la cible 'Type_of_Respiratory_Allergy_IGE_Poll

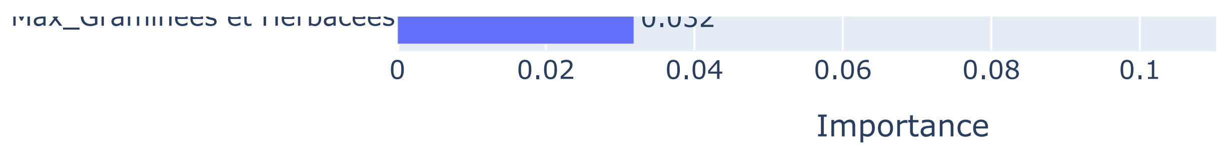


Top 10 Features pour la cible 'Type_of_Respiratory_Allergy_GINA' (X

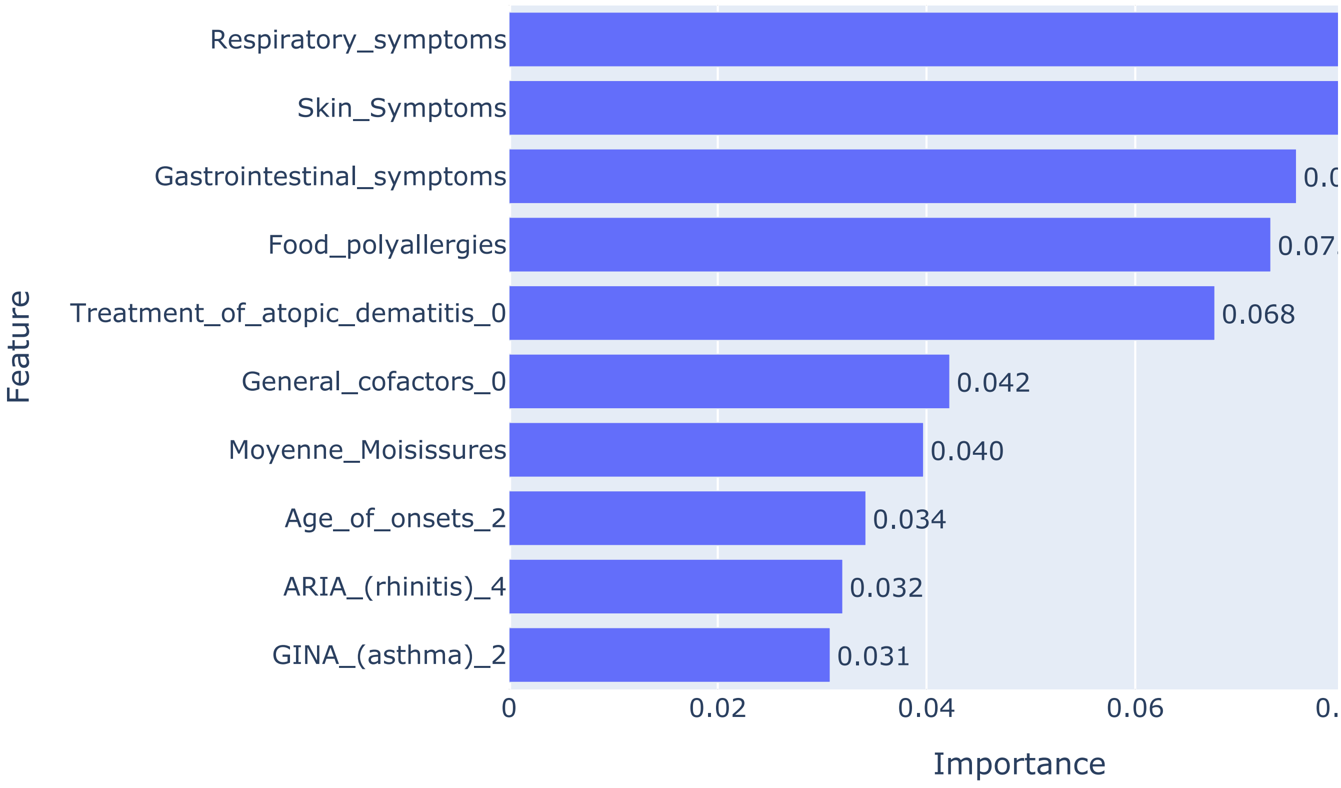


Top 10 Features pour la cible 'Type_of_Food_Allergy_Aromatics' (XG

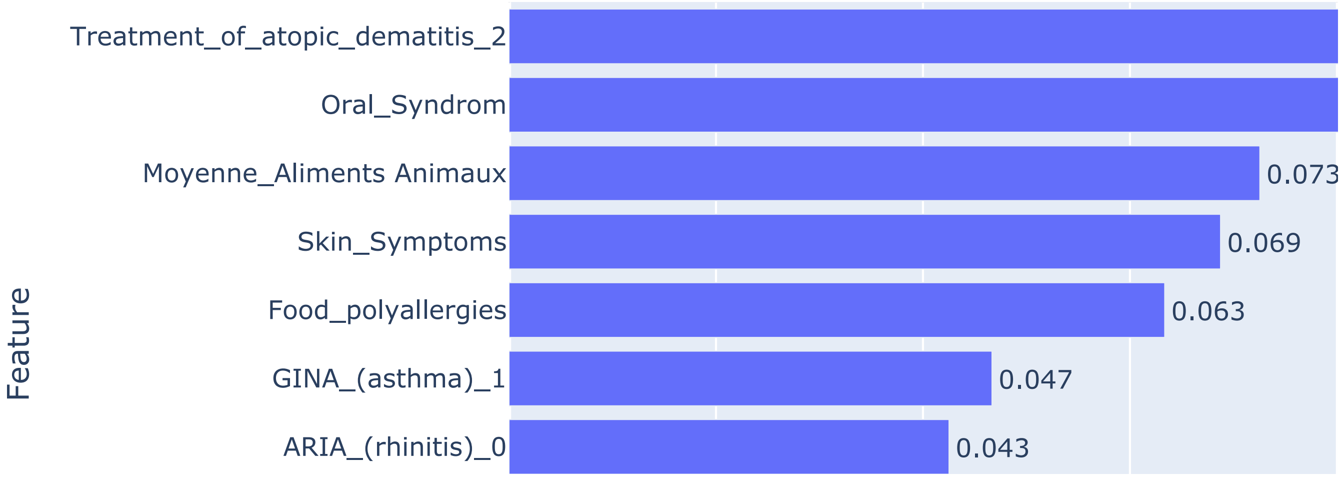


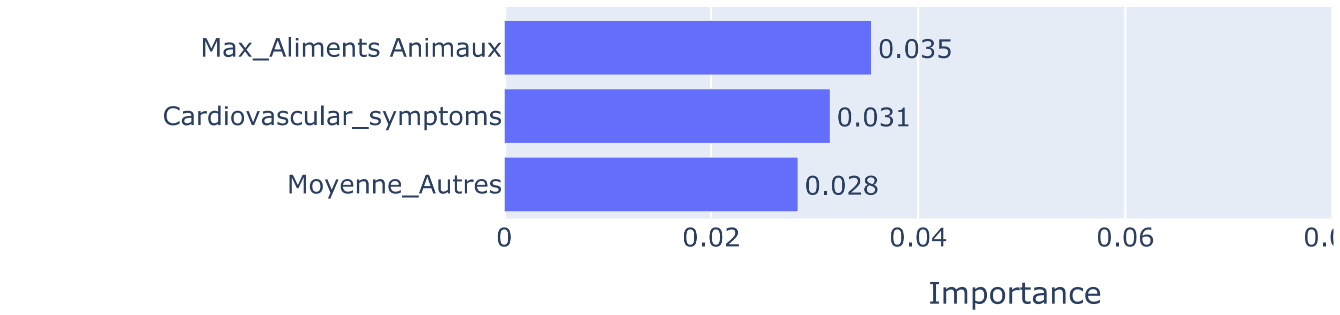


Top 10 Features pour la cible 'Type_of_Food_Allergy_Cereals_&_See

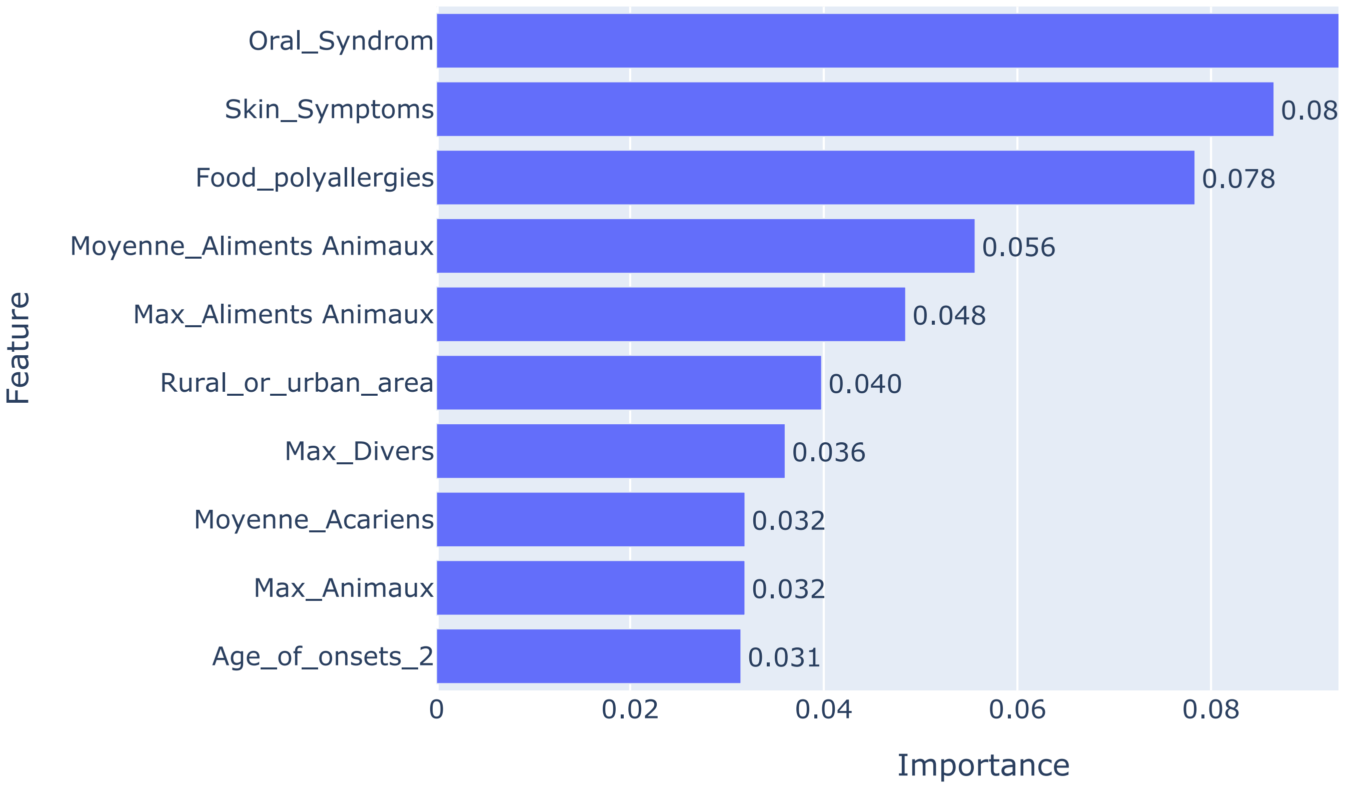


Top 10 Features pour la cible 'Type_of_Food_Allergy_Egg' (XGBoost)

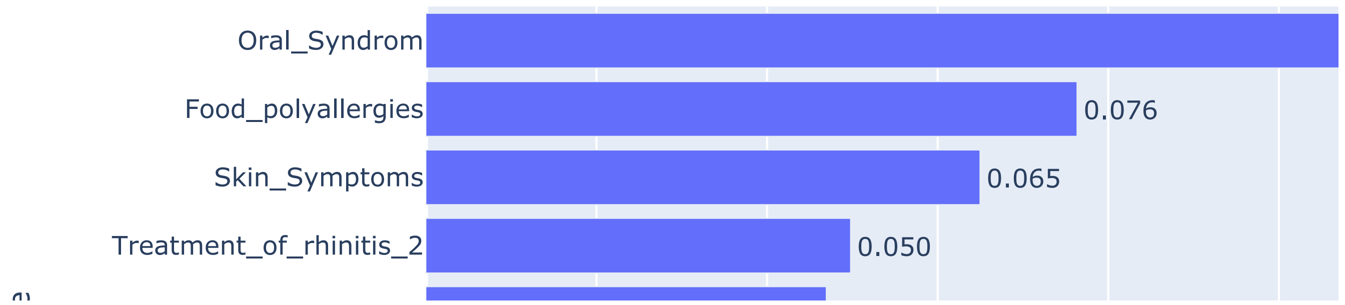


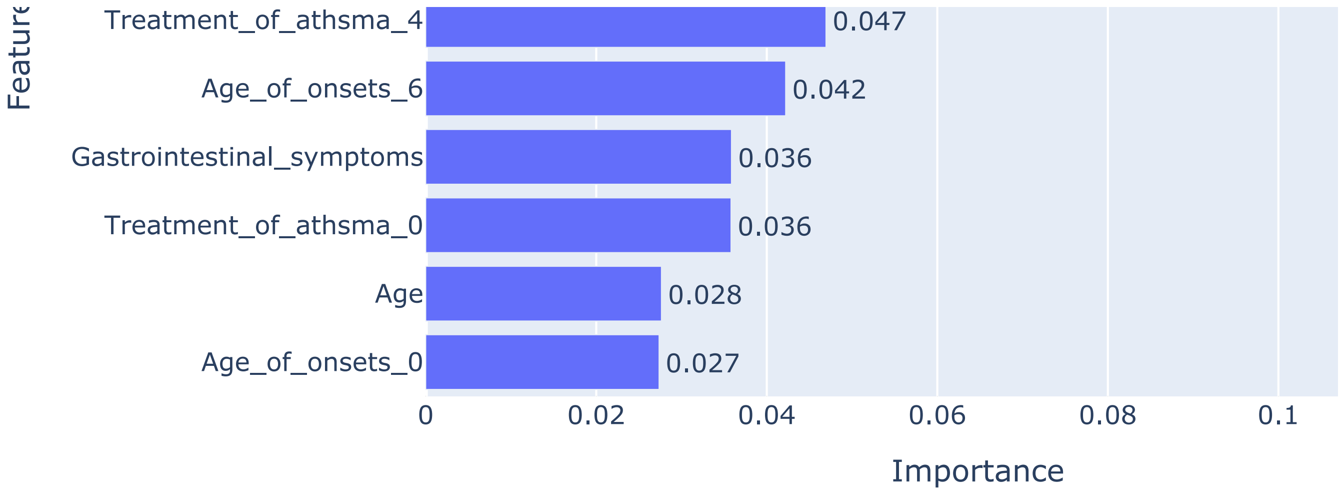


Top 10 Features pour la cible 'Type_of_Food_Allergy_Fish' (XGBoost)

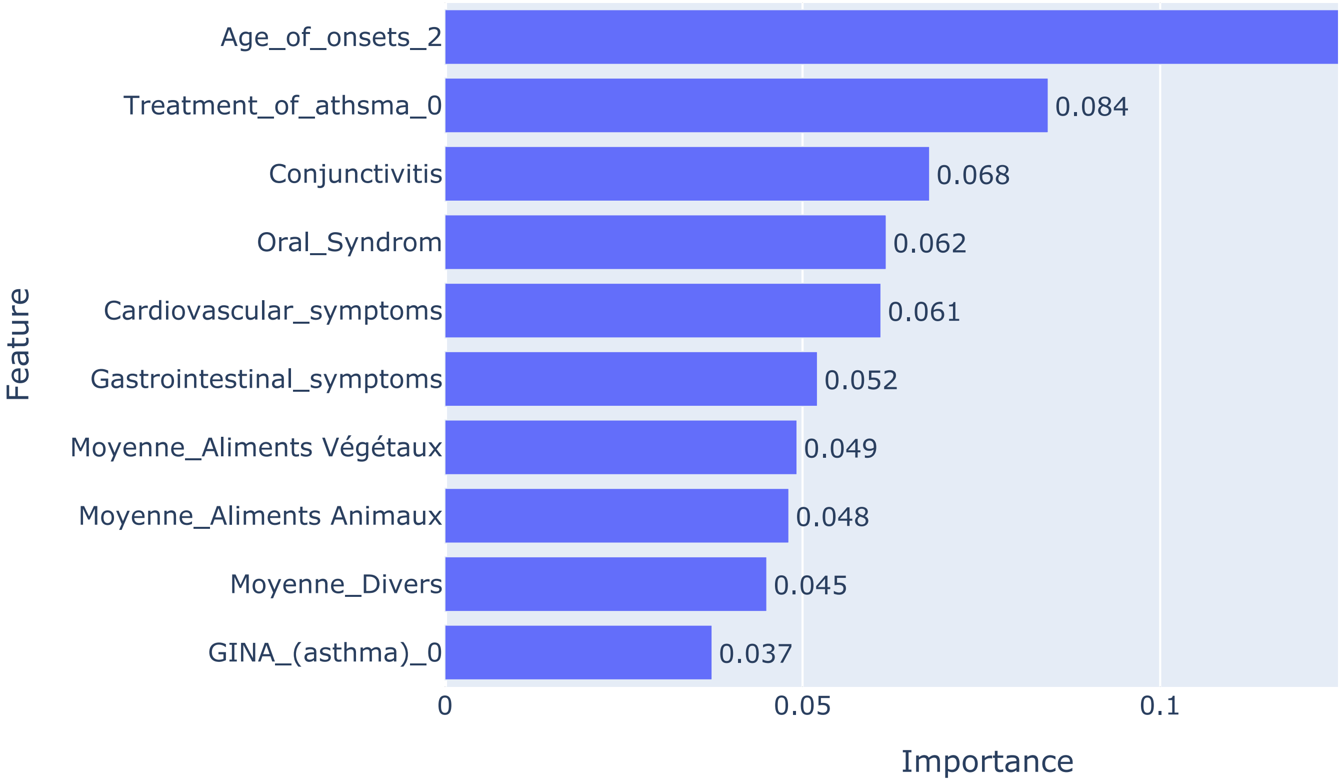


Top 10 Features pour la cible 'Type_of_Food_Allergy_Fruits_and_Veg

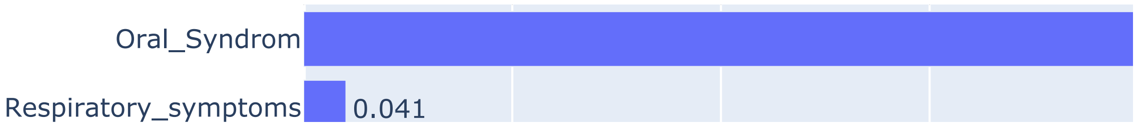


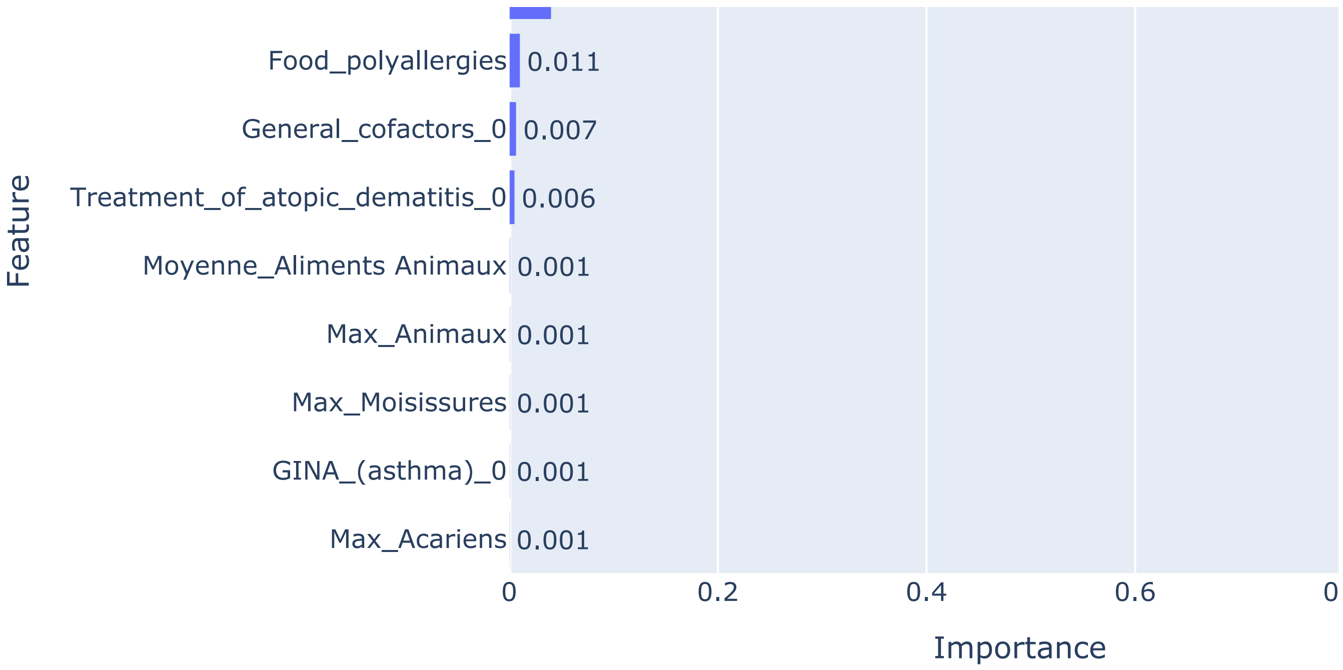


Top 10 Features pour la cible 'Type_of_Food_Allergy_Mammalian_Mi

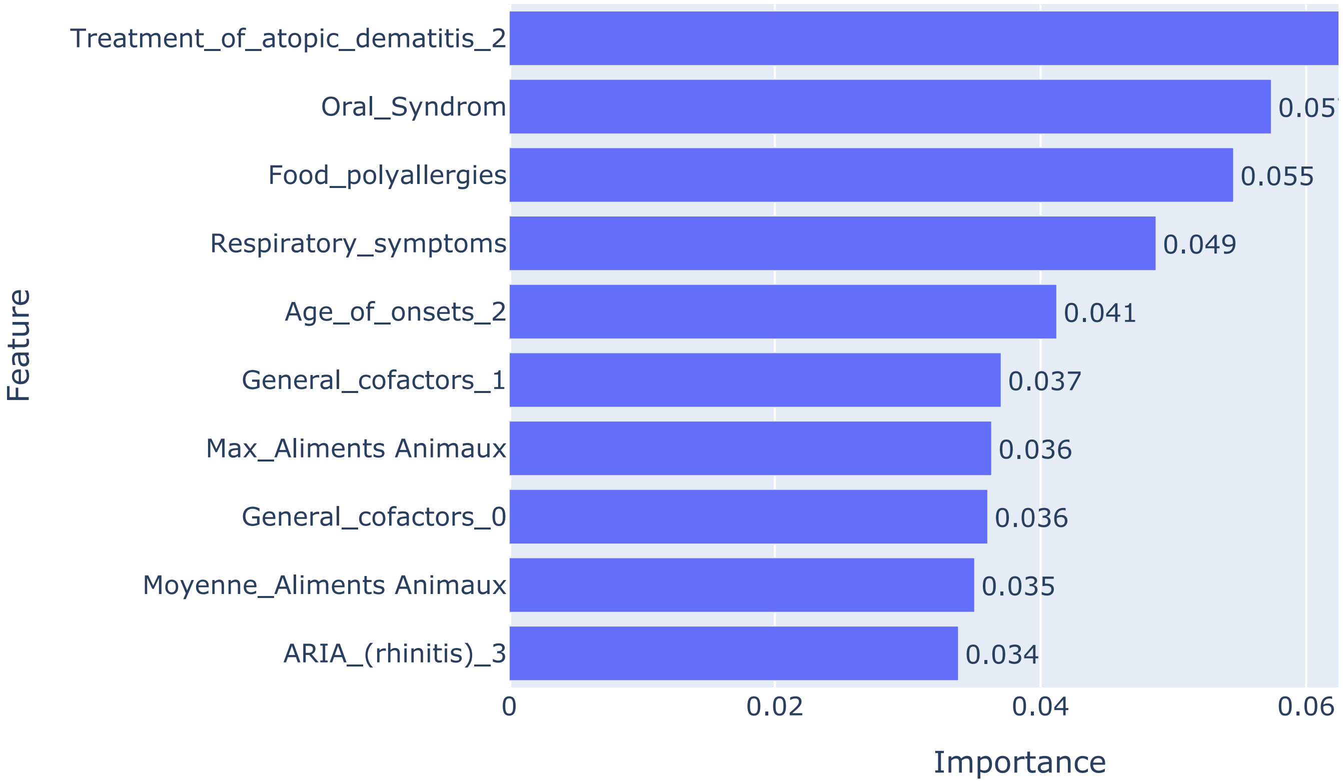


Top 10 Features pour la cible 'Type_of_Food_Allergy_Oral_Syndrom'

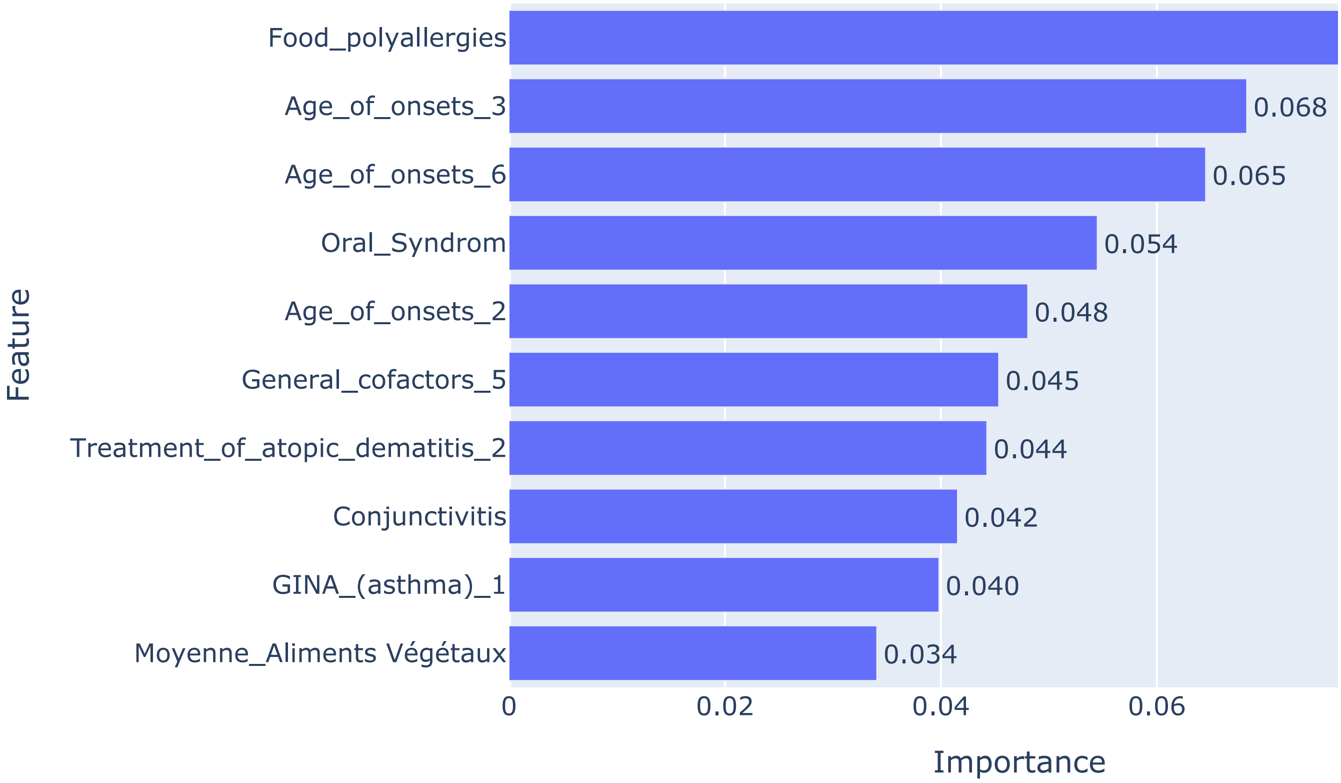




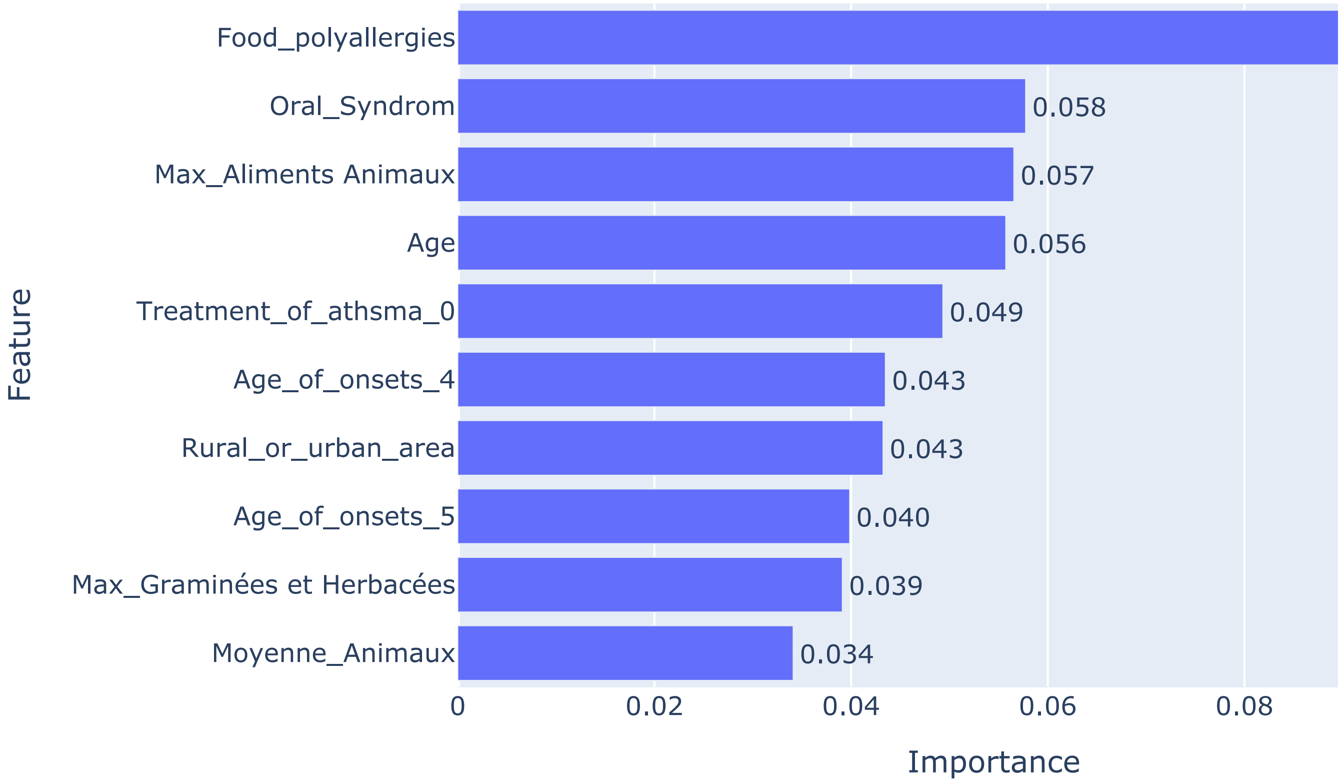
Top 10 Features pour la cible 'Type_of_Food_Allergy_Other_Legume



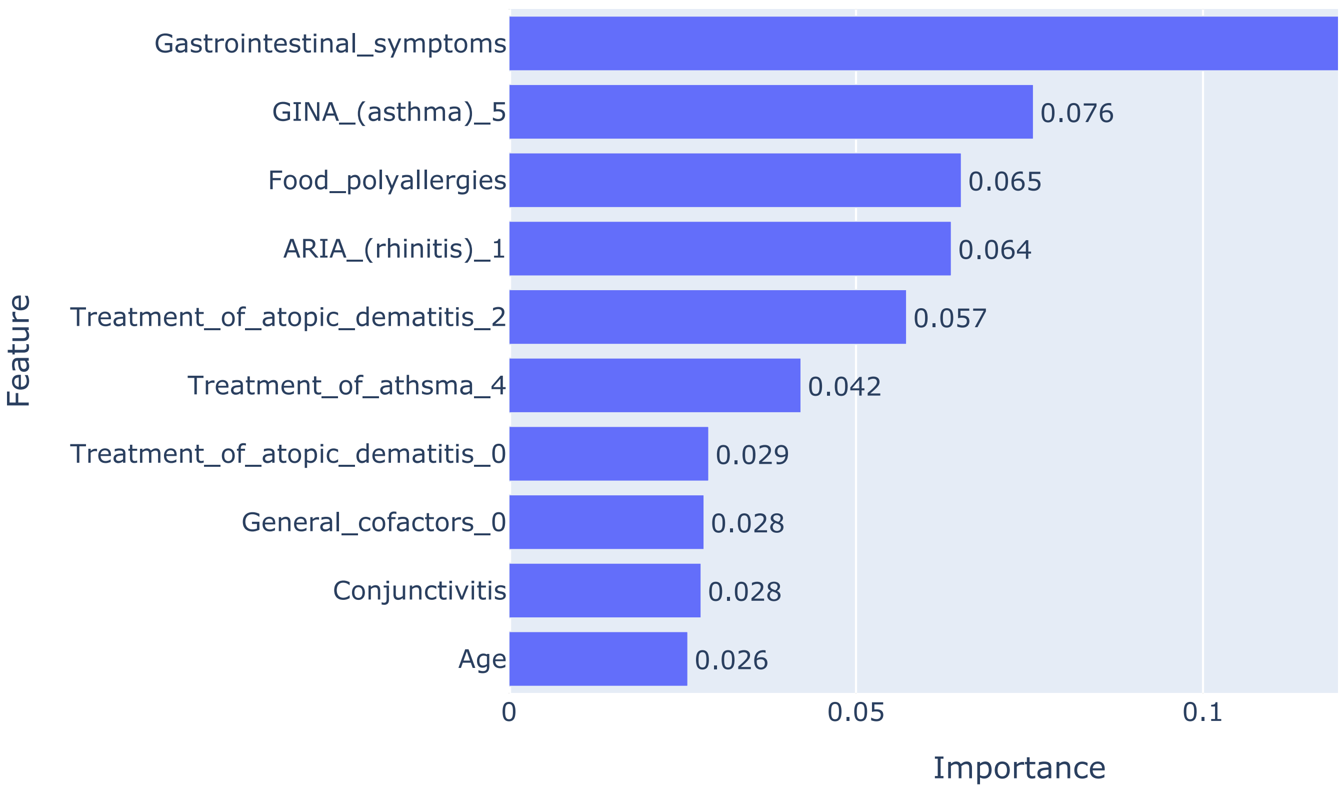
Top 10 Features pour la cible 'Type_of_Food_Allergy_Peanut' (XGBoo



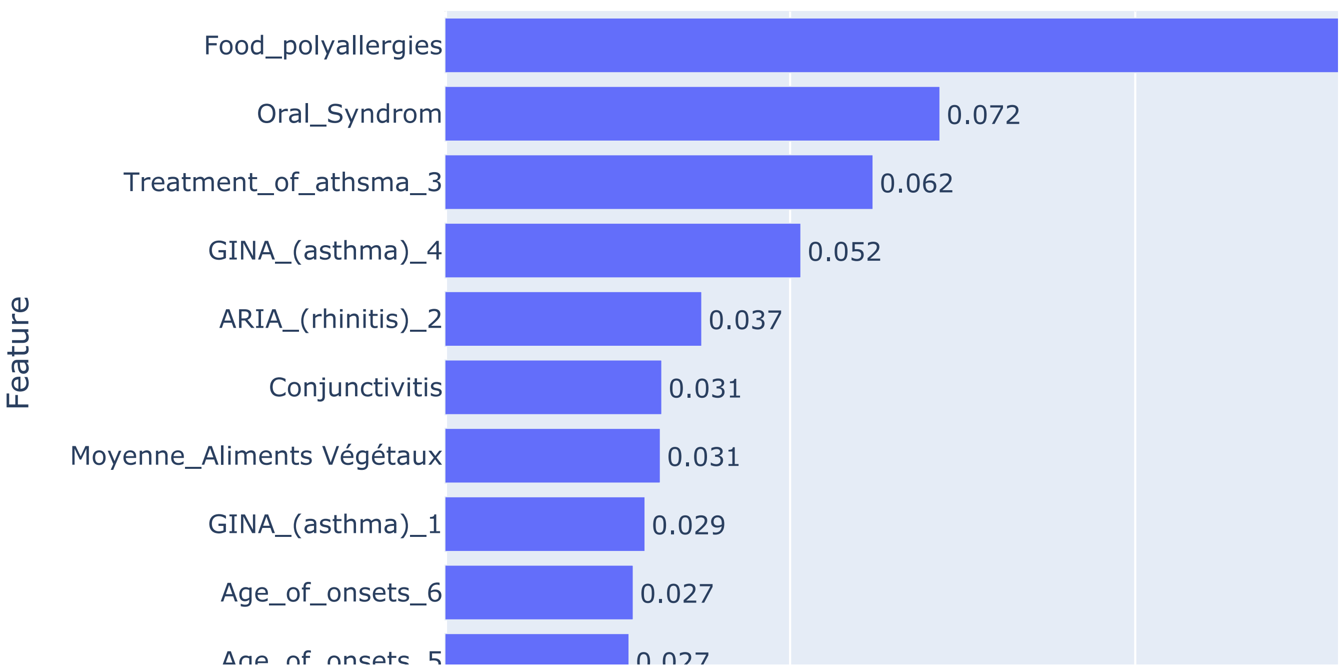
Top 10 Features pour la cible 'Type_of_Food_Allergy_Shellfish' (XGB

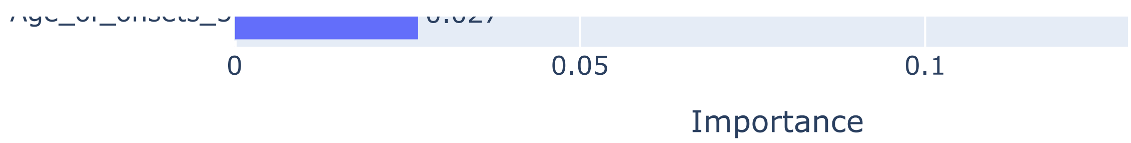


Top 10 Features pour la cible 'Type_of_Food_Allergy_TPO' (XGBoost)

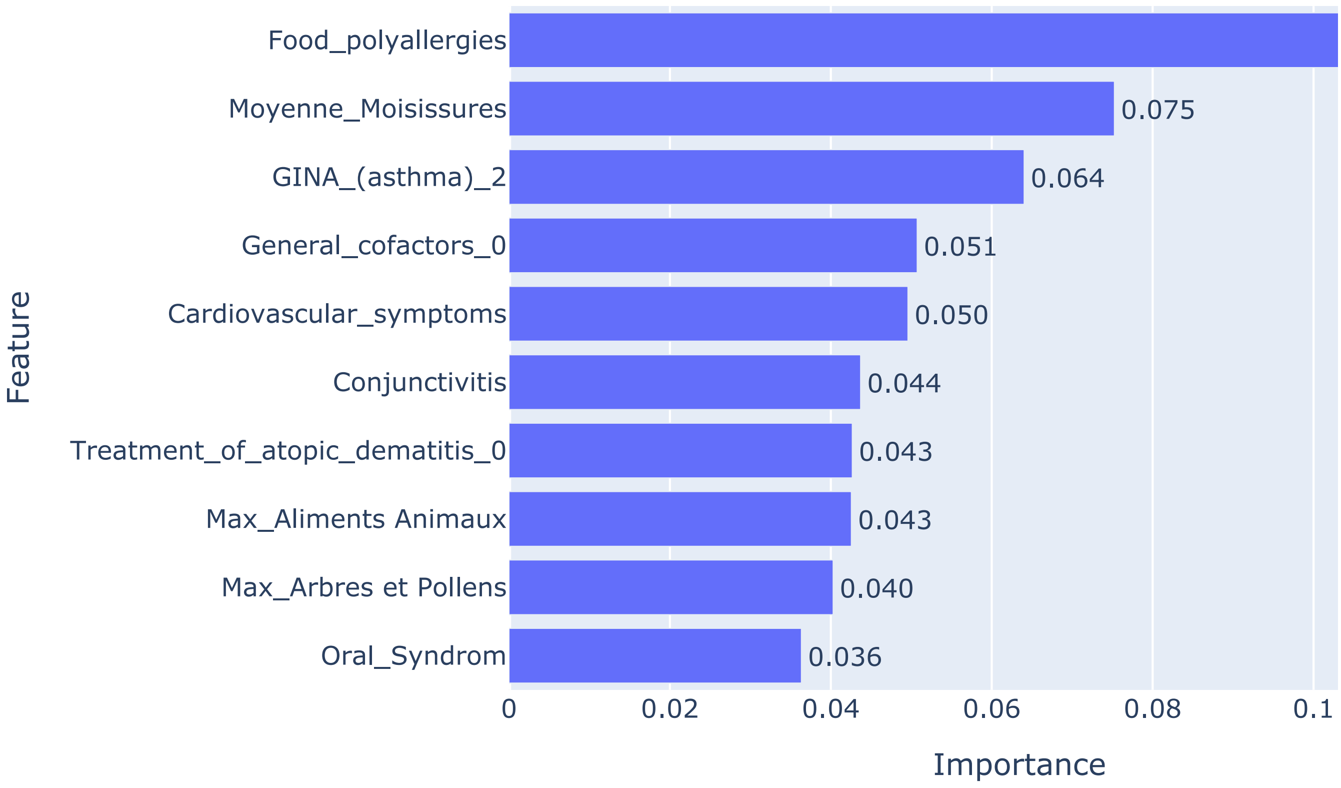


Top 10 Features pour la cible 'Type_of_Food_Allergy_Tree_Nuts' (XG

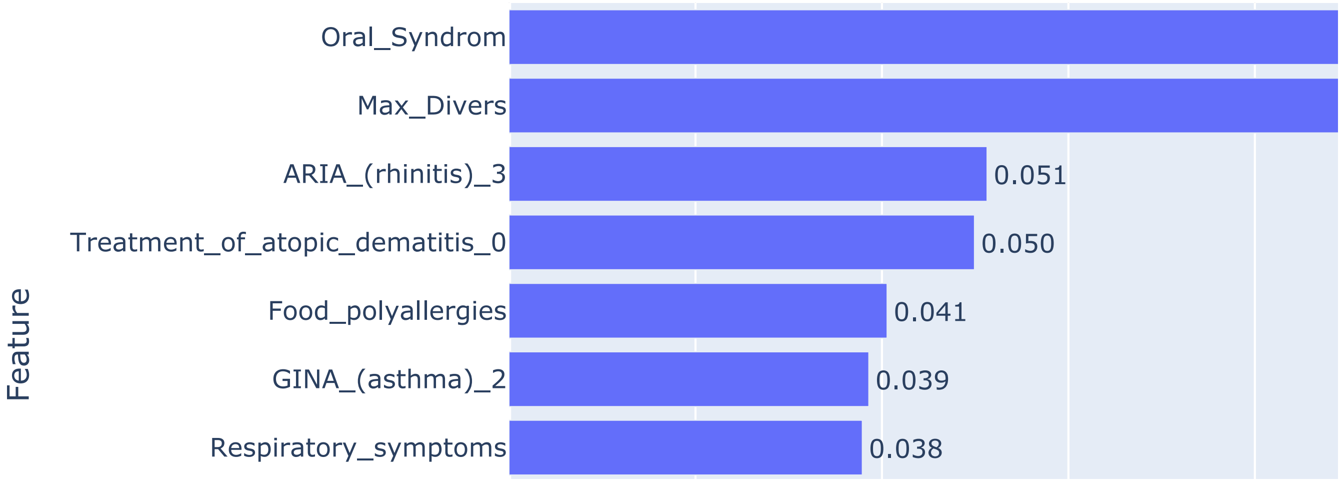


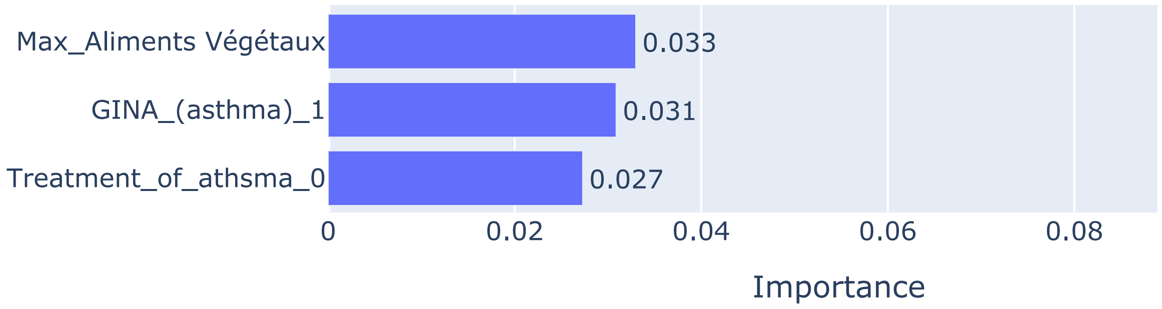


Top 10 Features pour la cible 'Type_of_Venom_Allergy_ATCD_Venon



Top 10 Features pour la cible 'Type_of_Venom_Allergy_IGE_Venom'





```
Start coding or generate with AI.
```