Imports

[ ] ↳ 2 cells hidden

Classification

```
import pandas as pd
V1 = pd.read_excel("ISAC_V1.xlsx", engine="openpyxl")
V1.head()
```

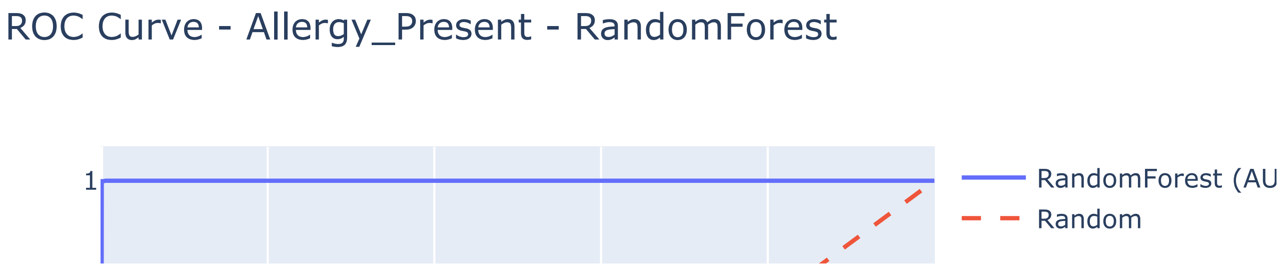
	Unnamed: 0	Chip_Code	Chip_Type	Age	Blood_Month_sample	French_Residence_
0	XPW0007	XPW0007	ISAC_V1	9	3	
1	XPW0011	XPW0011	ISAC_V1	25	5	
2	XPW0013	XPW0013	ISAC_V1	59	5	
3	XPW0017	XPW0017	ISAC_V1	49	5	
4	XPW0018	XPW0018	ISAC_V1	9	5	

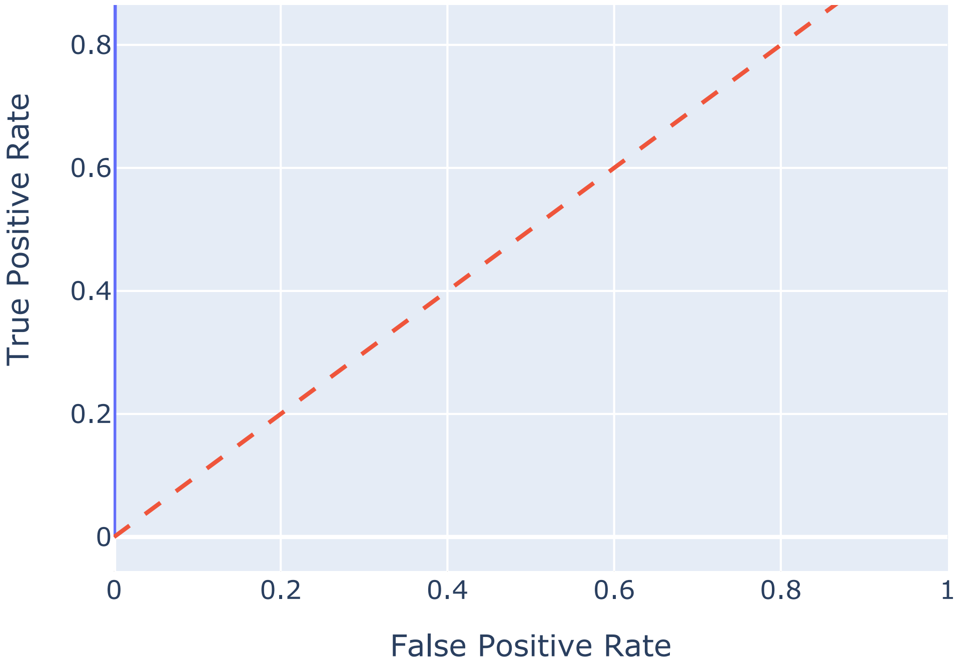
5 rows × 129 columns

```
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"
    ]

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", "Respi

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


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
```

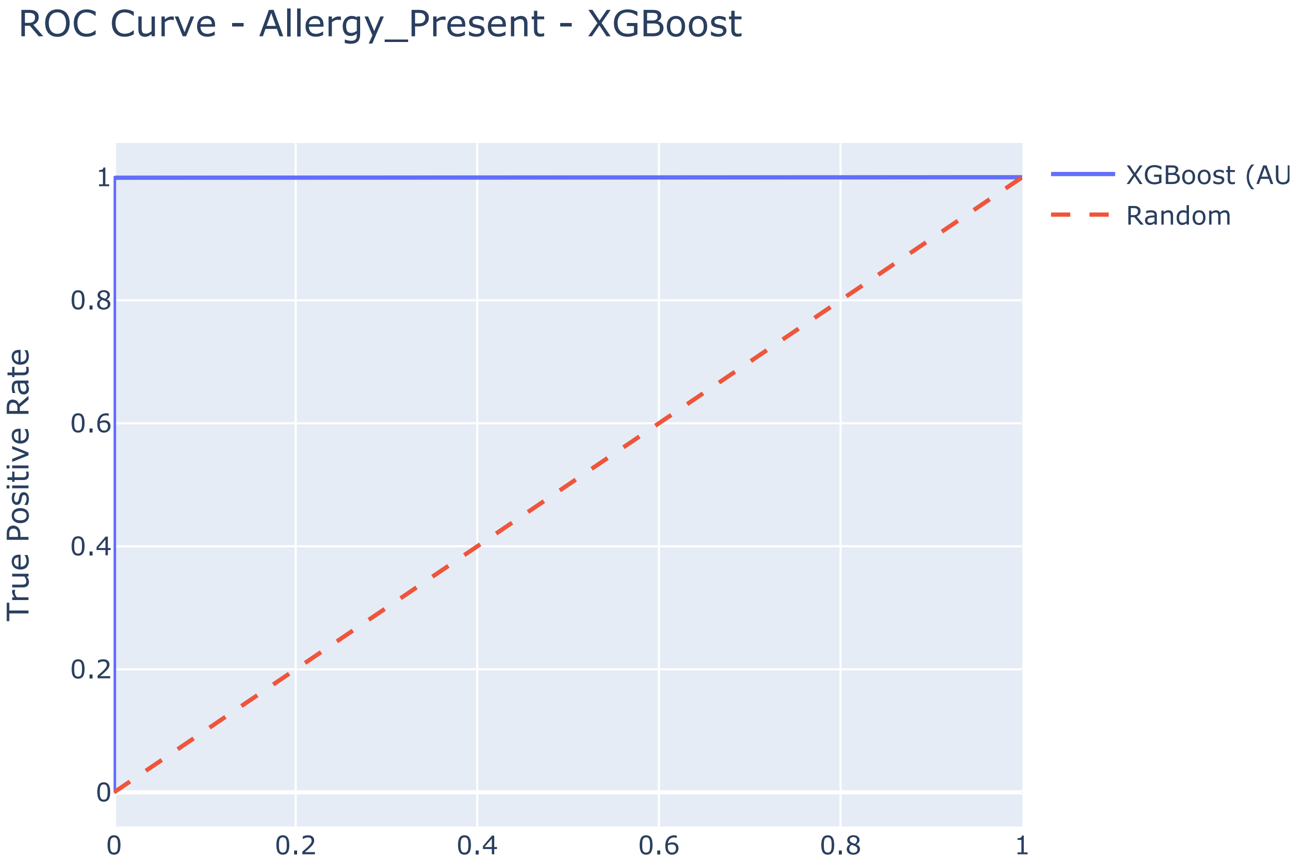
```
models = {
    "RandomForest": RandomForestClassifier(random_state=42),
    "XGBoost": XGBClassifier(random_state=42, eval_metric="logloss", use_label_
    "LogisticRegression": LogisticRegression(max_iter=1000, random_state=42),
    "SVM": SVC(probability=True, random_state=42)
}

X = V1.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 = V1[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.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_V1_Allergie.csv", index=False)
```

```
🔍 Target: Allergy_Present | Model: RandomForest
📈 Accuracy: 0.9426
🎯 F1 (0): 0.9400 | F1 (1): 0.9449
📊 Precision: 0.9432 | AUC: 0.981794232370263
📊 Confusion Matrix:
 [[1121    0]
 [   2 1228]]
```



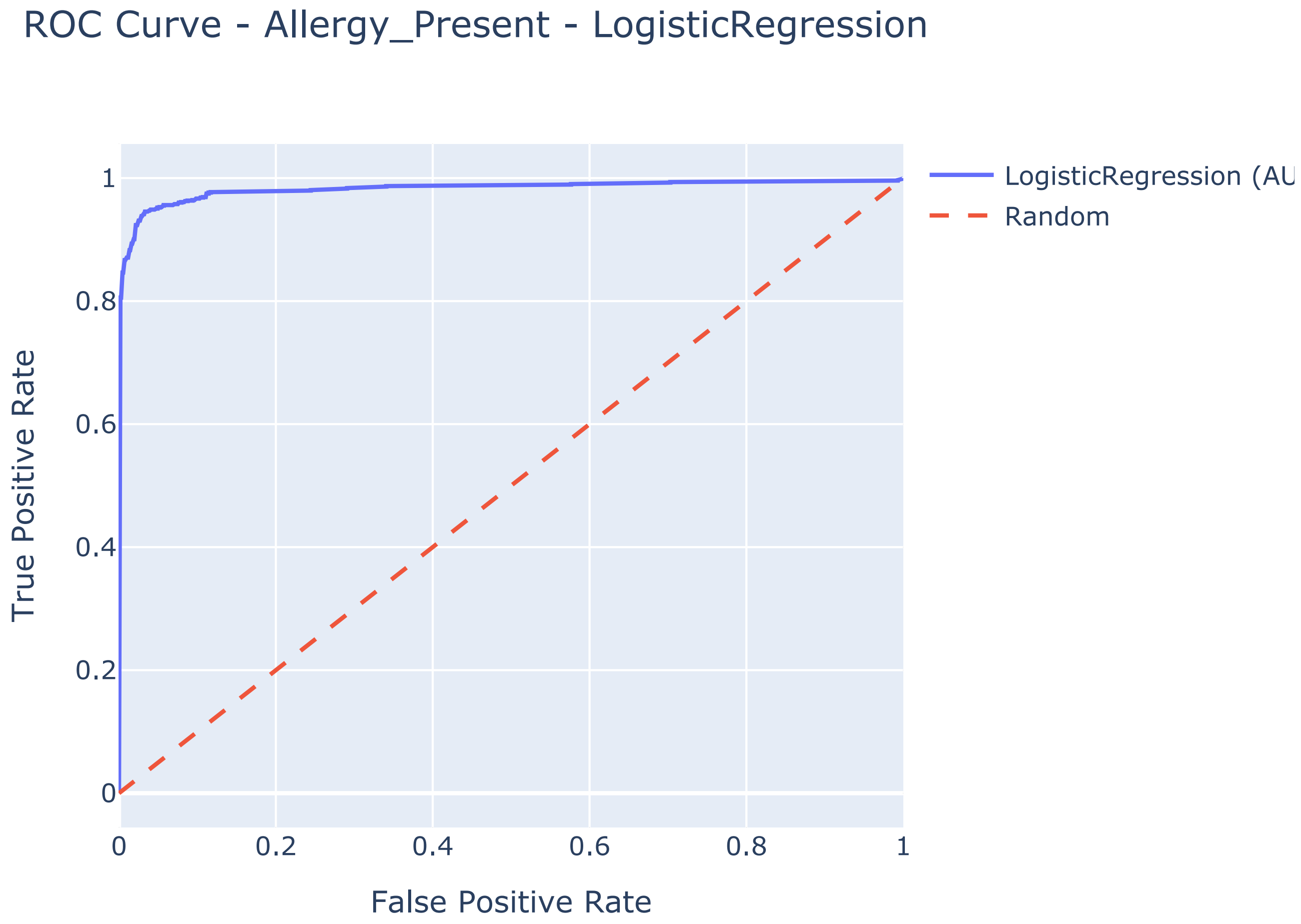


```
🔍 Target: Allergy_Present | Model: XGBoost
📈 Accuracy: 0.9400
🎯 F1 (0): 0.9386 | F1 (1): 0.9414
📊 Precision: 0.9414 | AUC: 0.975914184473703
📊 Confusion Matrix:
 [[1121    0]
 [   3 1227]]
```



False Positive Rate

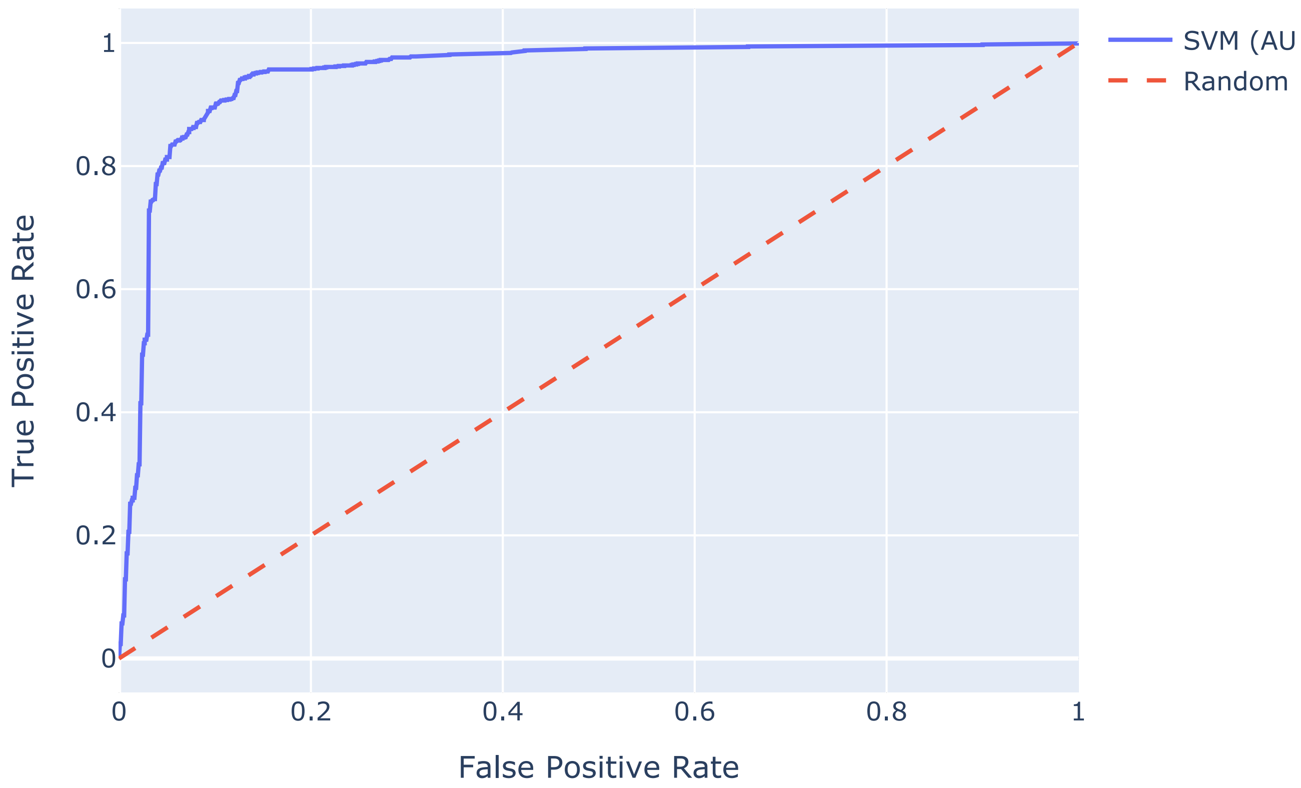
```
🔍 Target: Allergy_Present | Model: LogisticRegression
📈 Accuracy: 0.9409
🎯 F1 (0): 0.9402 | F1 (1): 0.9415
📊 Precision: 0.9437 | AUC: 0.9786801851109536
📊 Confusion Matrix:
 [[1095   26]
 [  87 1143]]
```



ROC Curve - Allergy_Present - LogisticRegression

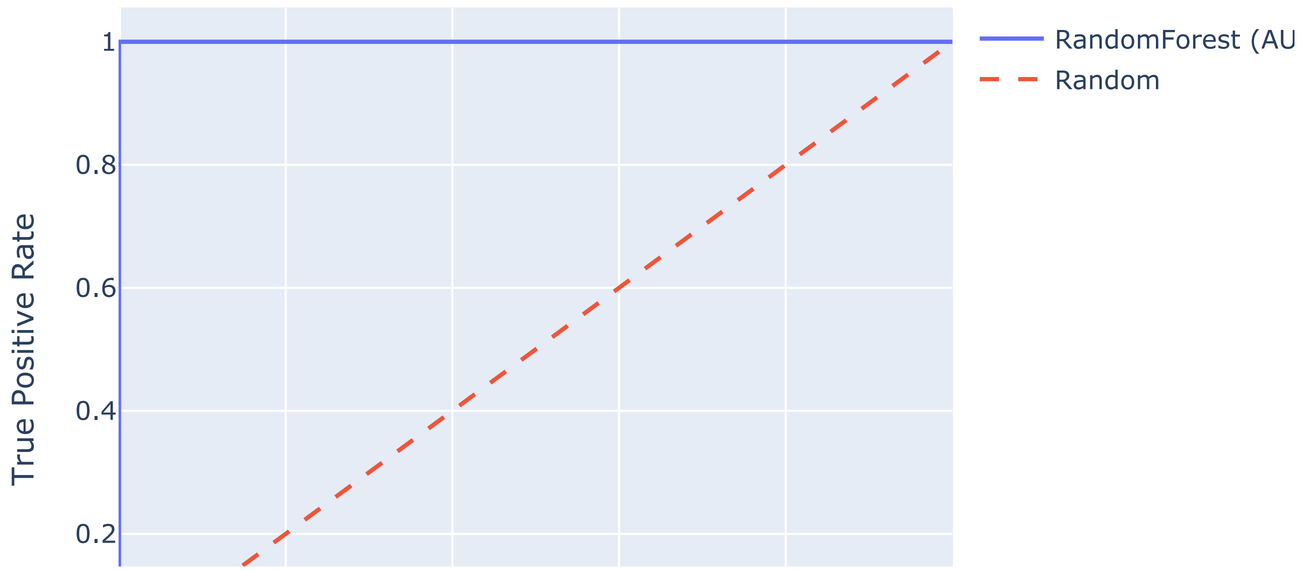
```
🔍 Target: Allergy_Present | Model: SVM
📈 Accuracy: 0.8532
🎯 F1 (0): 0.8598 | F1 (1): 0.8458
📊 Precision: 0.8676 | AUC: 0.9411421556535414
📊 Confusion Matrix:
 [[1071   50]
 [ 246  984]]
```

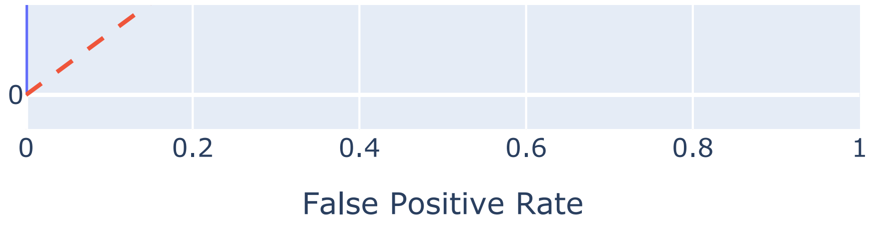
ROC Curve - Allergy_Present - SVM



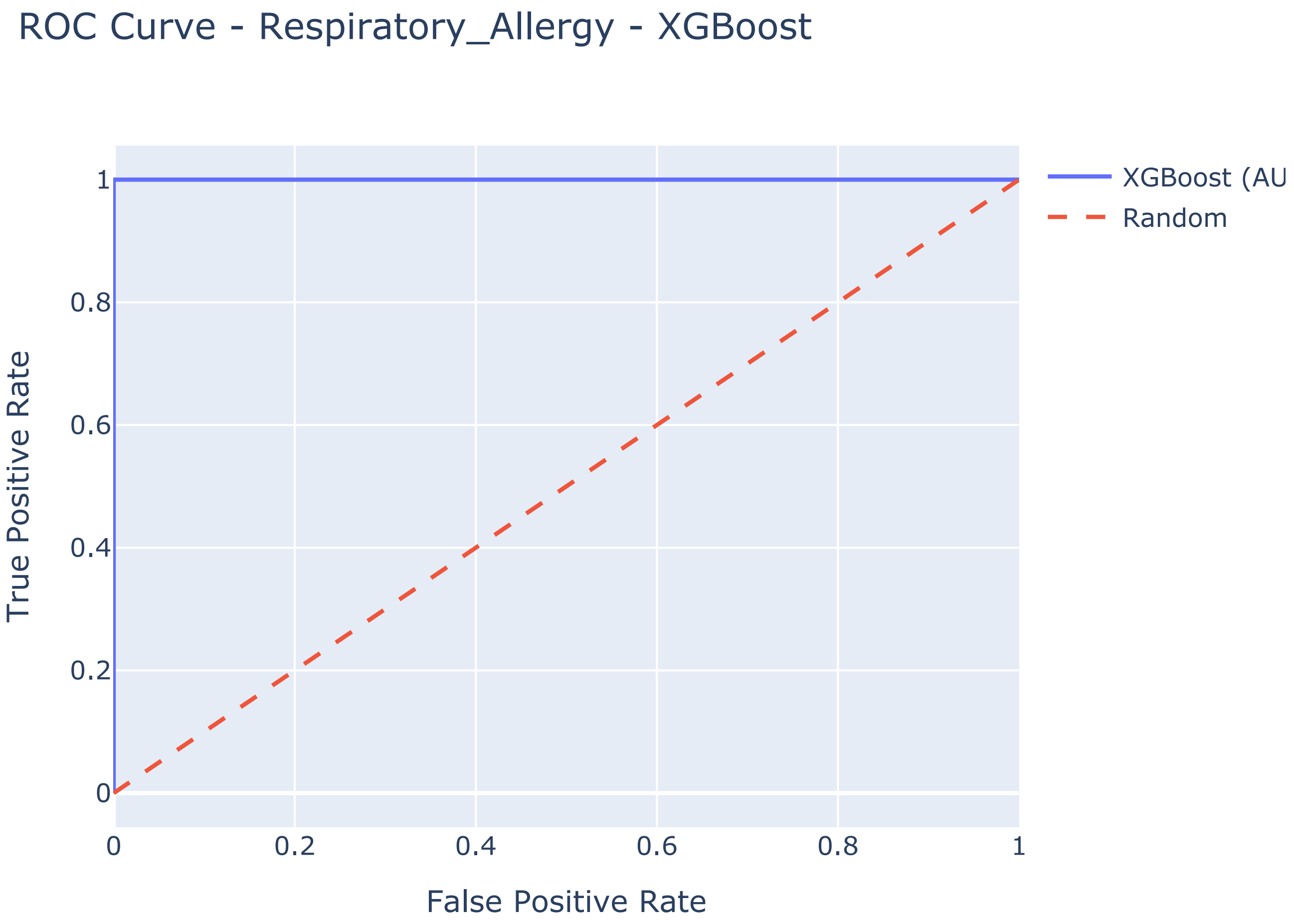
```
🔍 Target: Respiratory_Allergy | Model: RandomForest
📈 Accuracy: 0.9575
🎯 F1 (0): 0.9612 | F1 (1): 0.9530
📊 Precision: 0.9580 | AUC: 0.9869118890060473
📊 Confusion Matrix:
 [[1288    0]
 [   0 1063]]
```

ROC Curve - Respiratory_Allergy - RandomForest



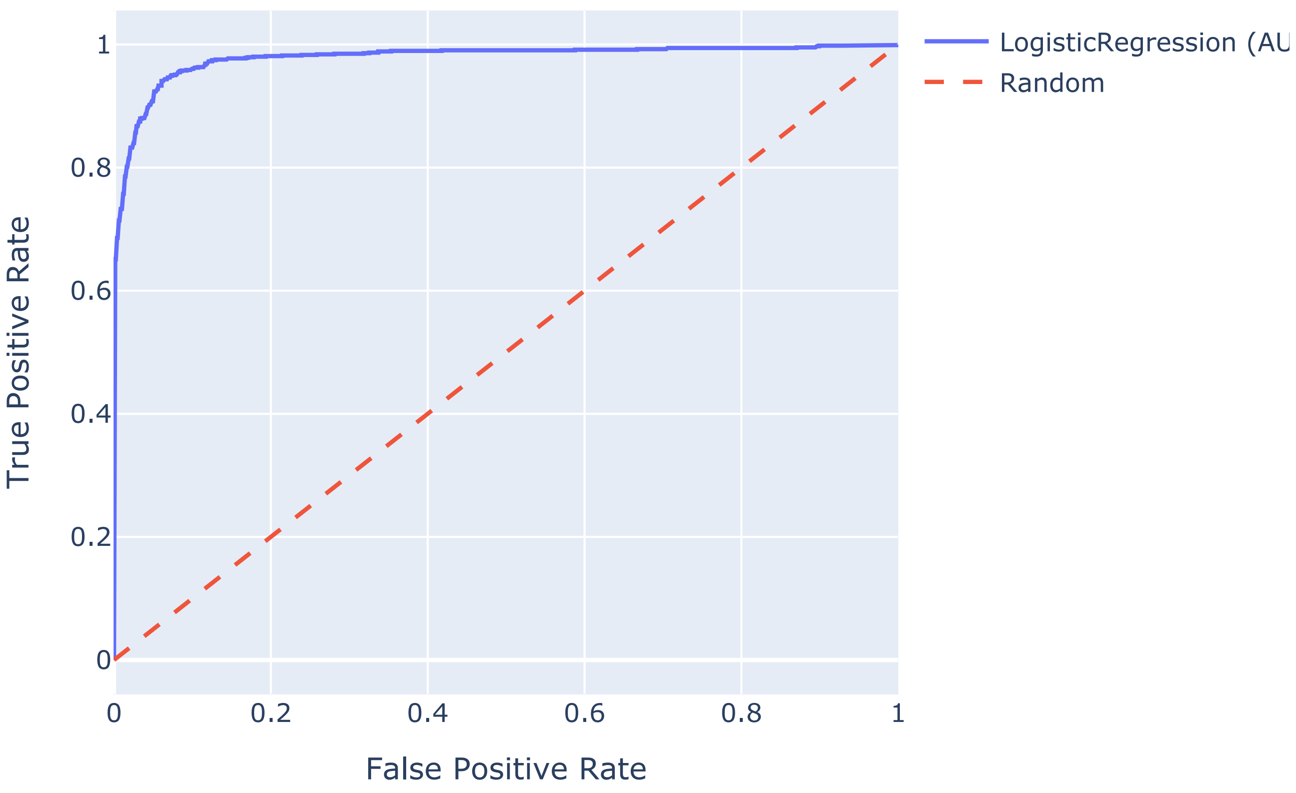


```
🔍 Target: Respiratory_Allergy | Model: XGBoost
📈 Accuracy: 0.9494
🎯 F1 (0): 0.9544 | F1 (1): 0.9431
📊 Precision: 0.9501 | AUC: 0.9854397135518118
📊 Confusion Matrix:
 [[1288    0]
 [   0 1063]]
```



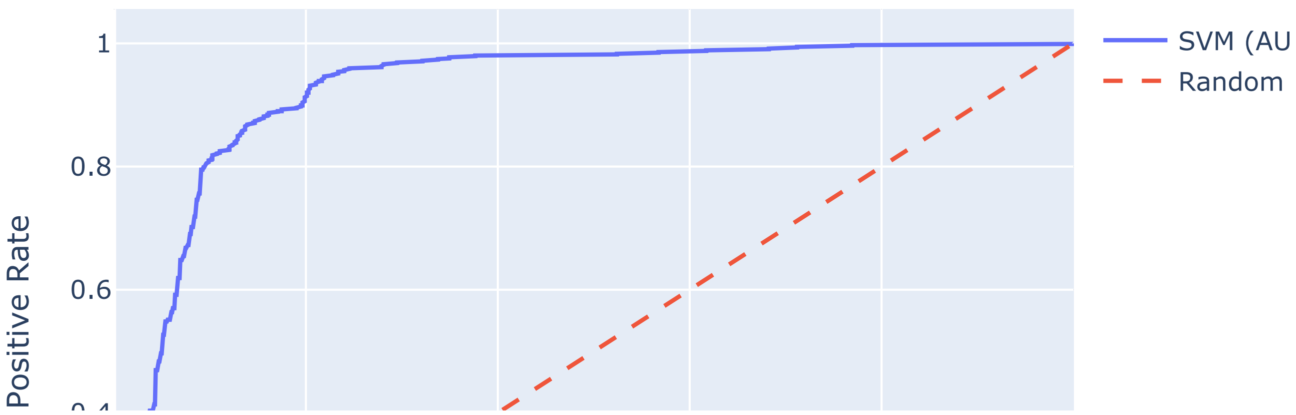
```
🔍 Target: Respiratory_Allergy | Model: LogisticRegression
📈 Accuracy: 0.9230
🎯 F1 (0): 0.9308 | F1 (1): 0.9132
📊 Precision: 0.9236 | AUC: 0.9700775202341848
📊 Confusion Matrix:
 [[1234   54]
 [ 117  946]]
```

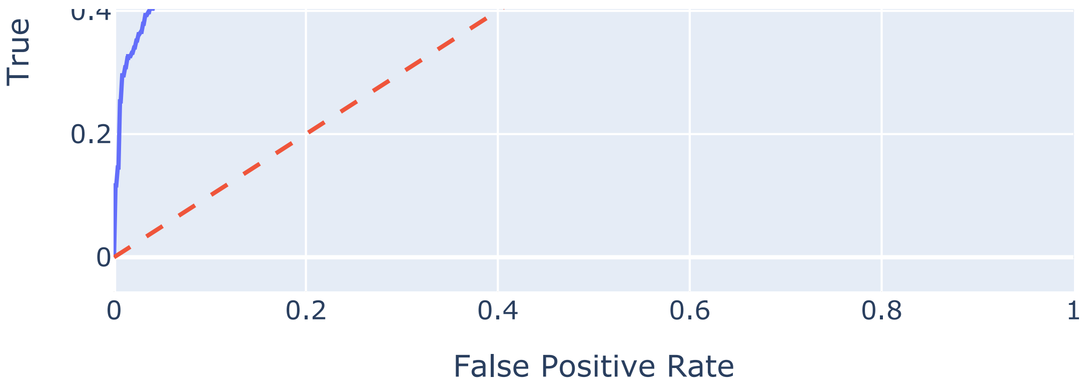
ROC Curve - Respiratory_Allergy - LogisticRegression



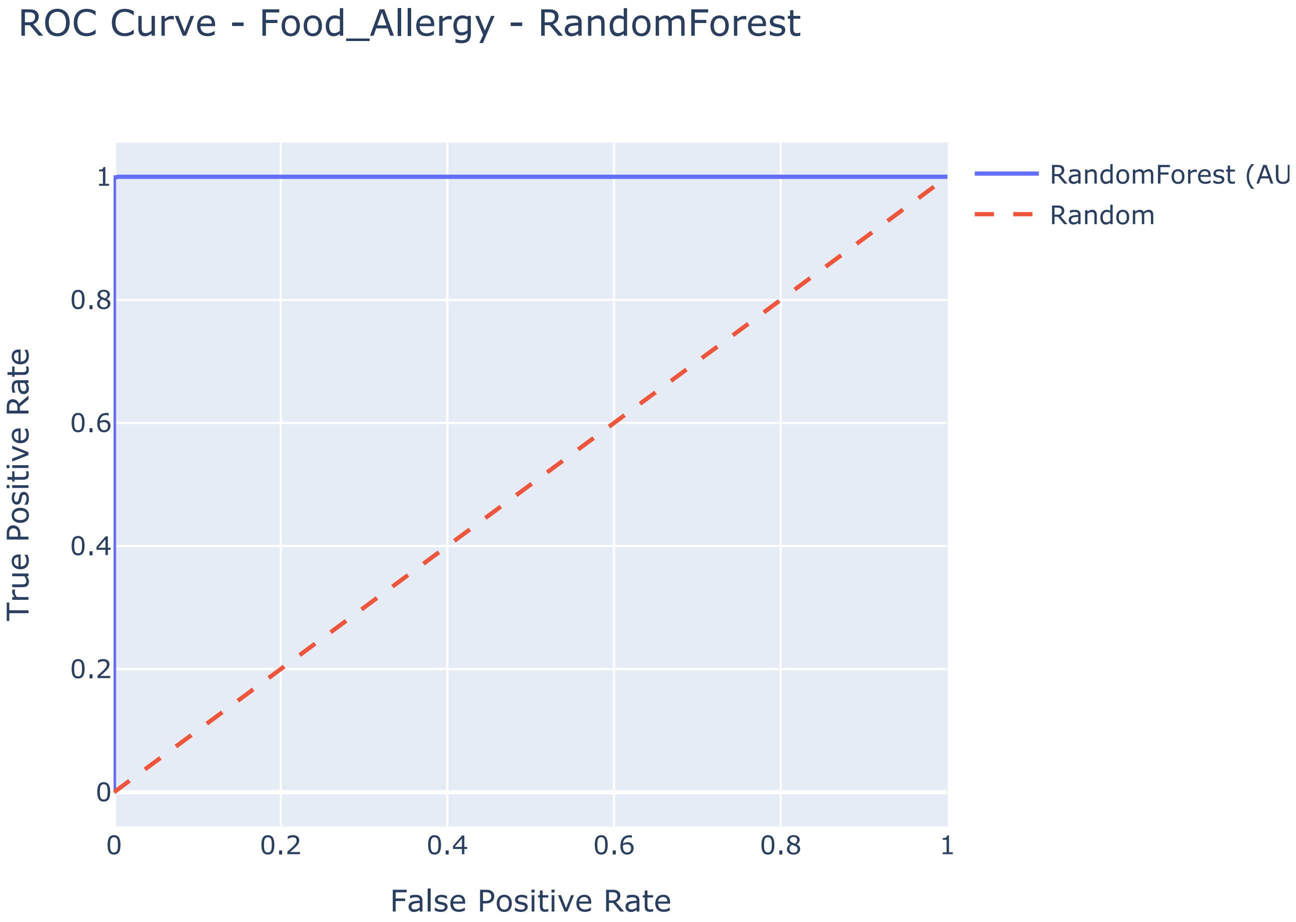
```
🔍 Target: Respiratory_Allergy | Model: SVM
📈 Accuracy: 0.8350
🎯 F1 (0): 0.8531 | F1 (1): 0.8117
📊 Precision: 0.8354 | AUC: 0.9091027731529515
📊 Confusion Matrix:
 [[1166  122]
 [ 213  850]]
```

ROC Curve - Respiratory_Allergy - SVM



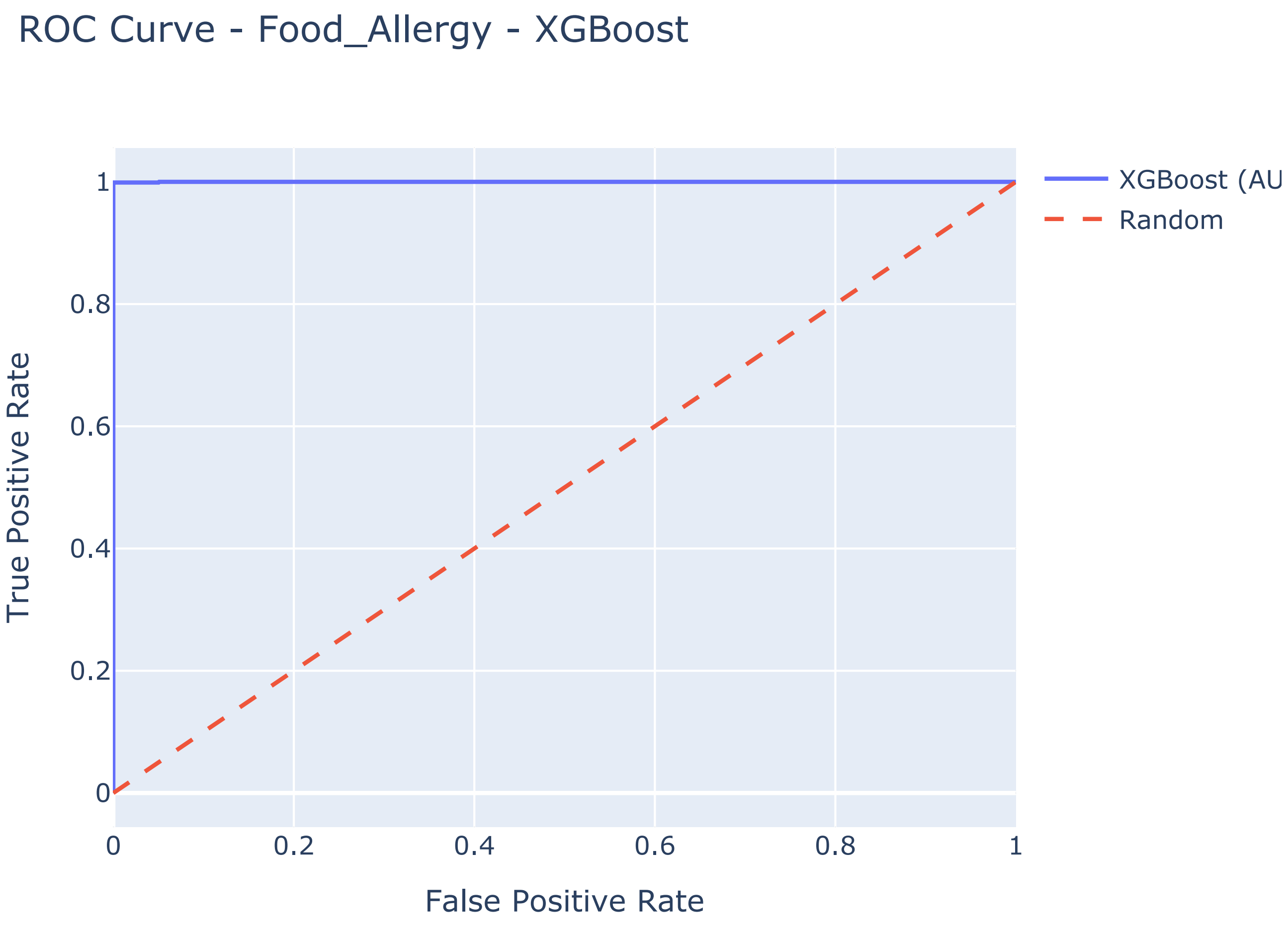


```
🔍 Target: Food_Allergy | Model: RandomForest
📈 Accuracy: 0.9039
🎯 F1 (0): 0.9223 | F1 (1): 0.8739
📊 Precision: 0.9044 | AUC: 0.962290753115054
📊 Confusion Matrix:
 [[1457    0]
 [   1  893]]
```

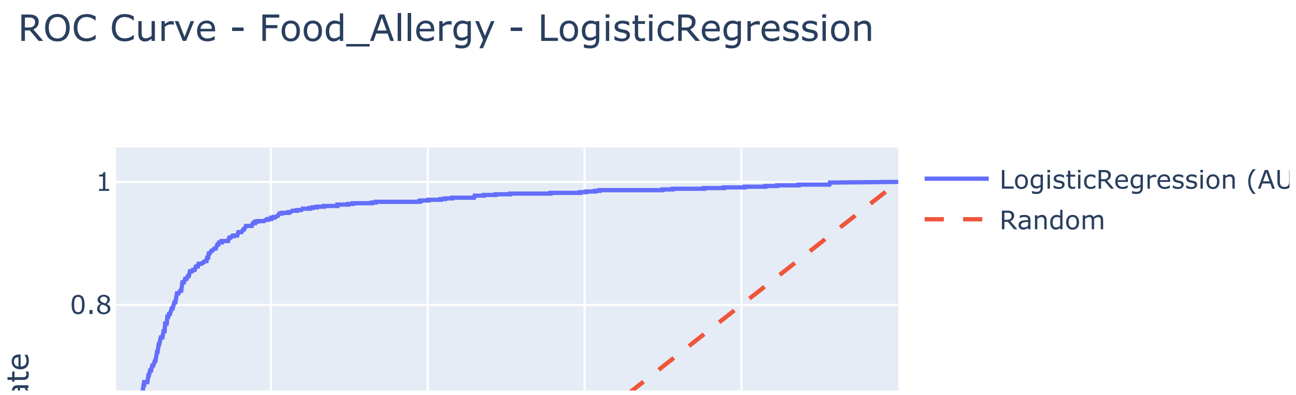


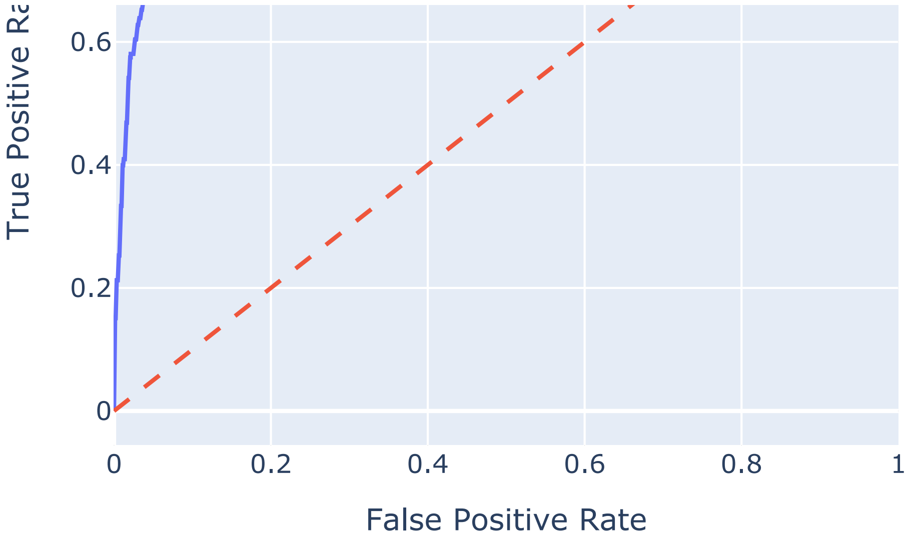
```
🔍 Target: Food_Allergy | Model: XGBoost
📈 Accuracy: 0.9009
🎯 F1 (0): 0.9208 | F1 (1): 0.8676
```

```
📊 Precision: 0.9008 | AUC: 0.9569938798750026
📊 Confusion Matrix:
 [[1457    0]
 [   1  893]]
```



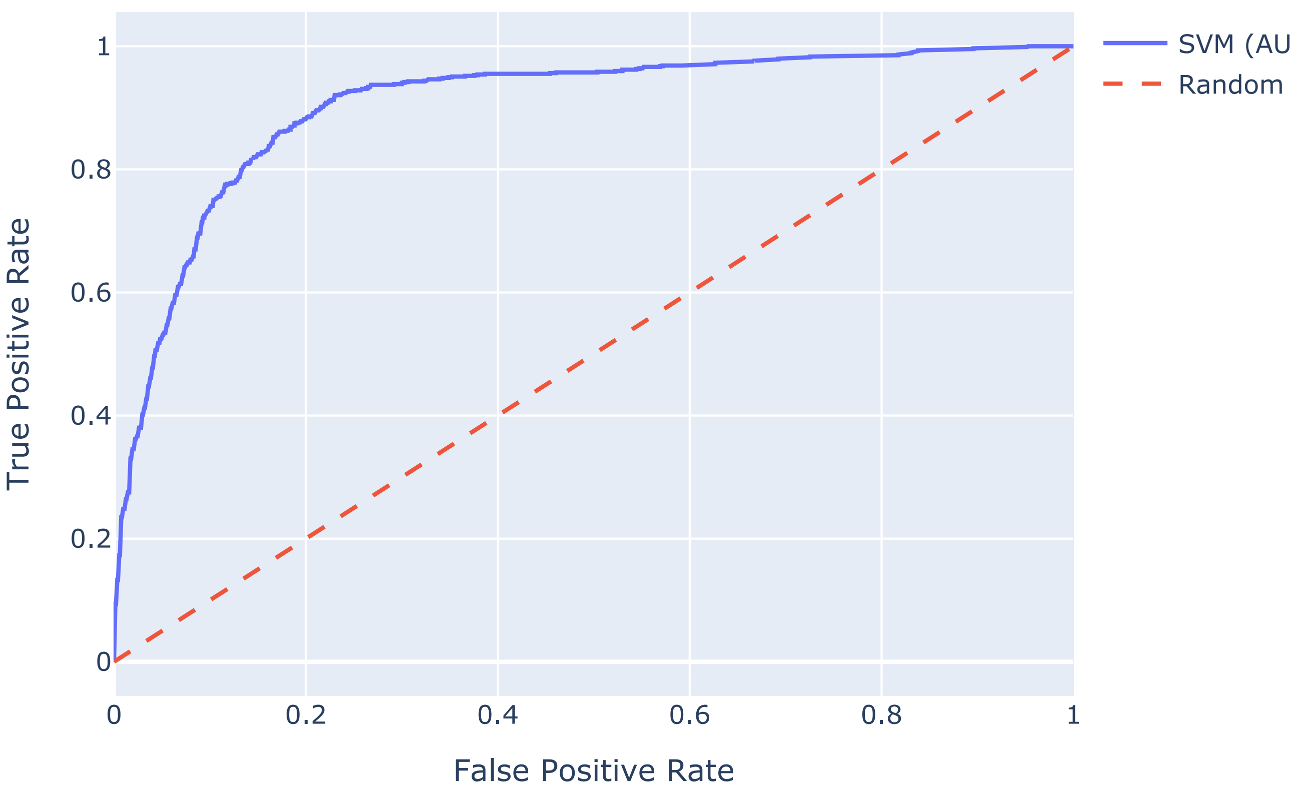
```
🔍 Target: Food_Allergy | Model: LogisticRegression
📈 Accuracy: 0.8622
🎯 F1 (0): 0.8886 | F1 (1): 0.8190
📊 Precision: 0.8634 | AUC: 0.9215525792334454
📊 Confusion Matrix:
 [[1346  111]
 [ 179  715]]
```



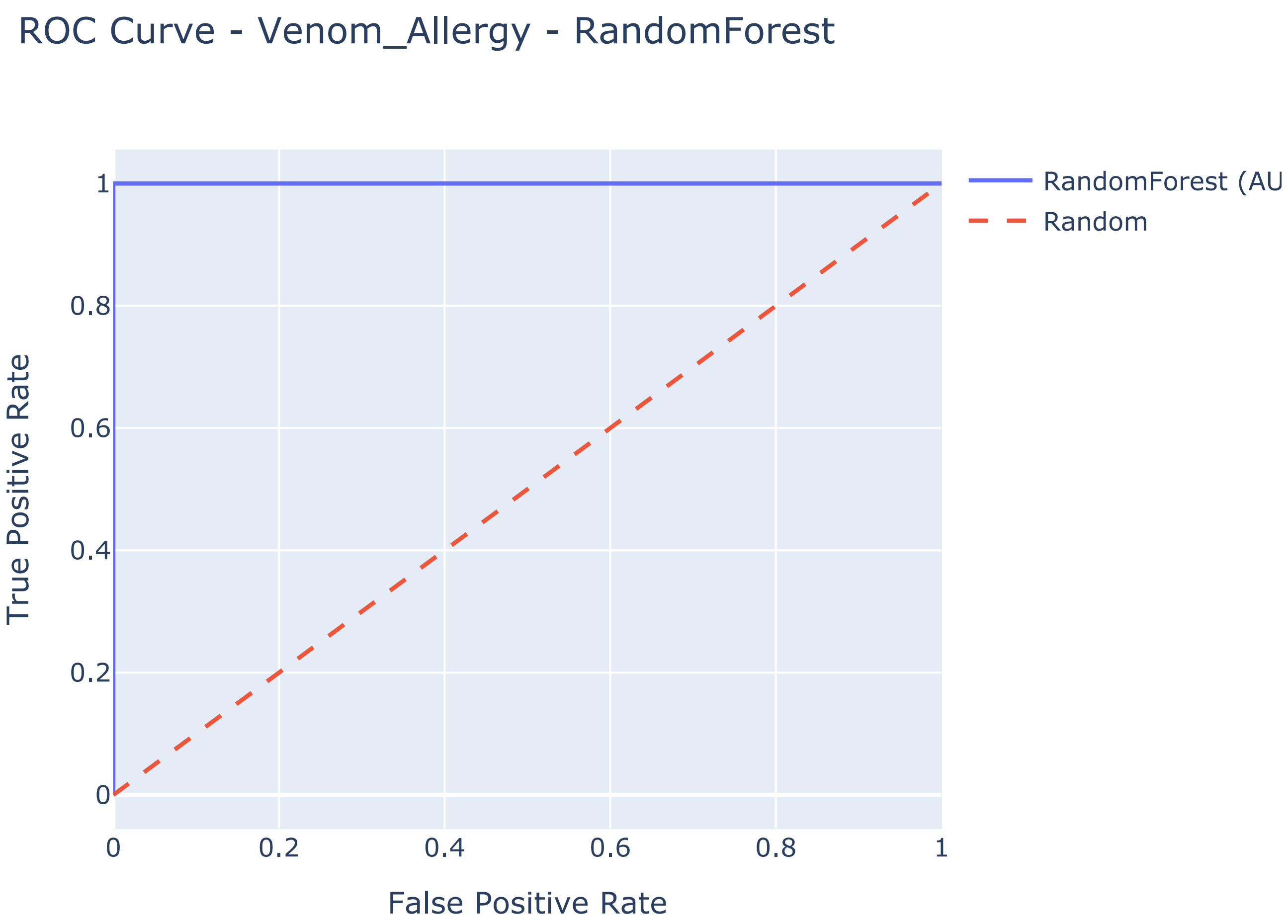


```
🔍 Target: Food_Allergy | Model: SVM
📈 Accuracy: 0.8103
🎯 F1 (0): 0.8453 | F1 (1): 0.7545
📊 Precision: 0.8121 | AUC: 0.8829788679361001
📊 Confusion Matrix:
 [[1324  133]
 [ 261  633]]
```

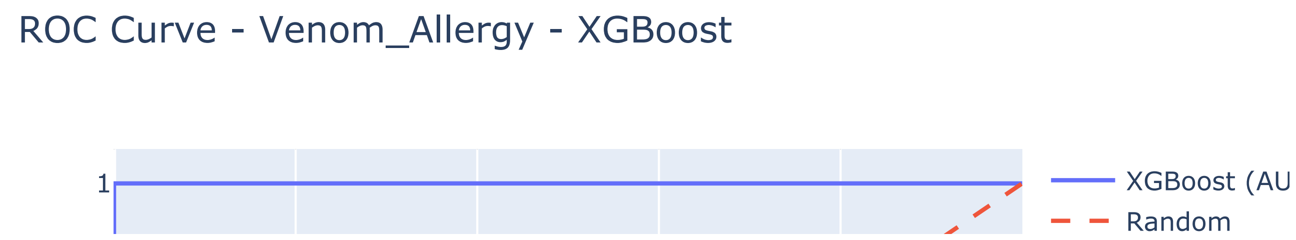
ROC Curve - Food_Allergy - SVM

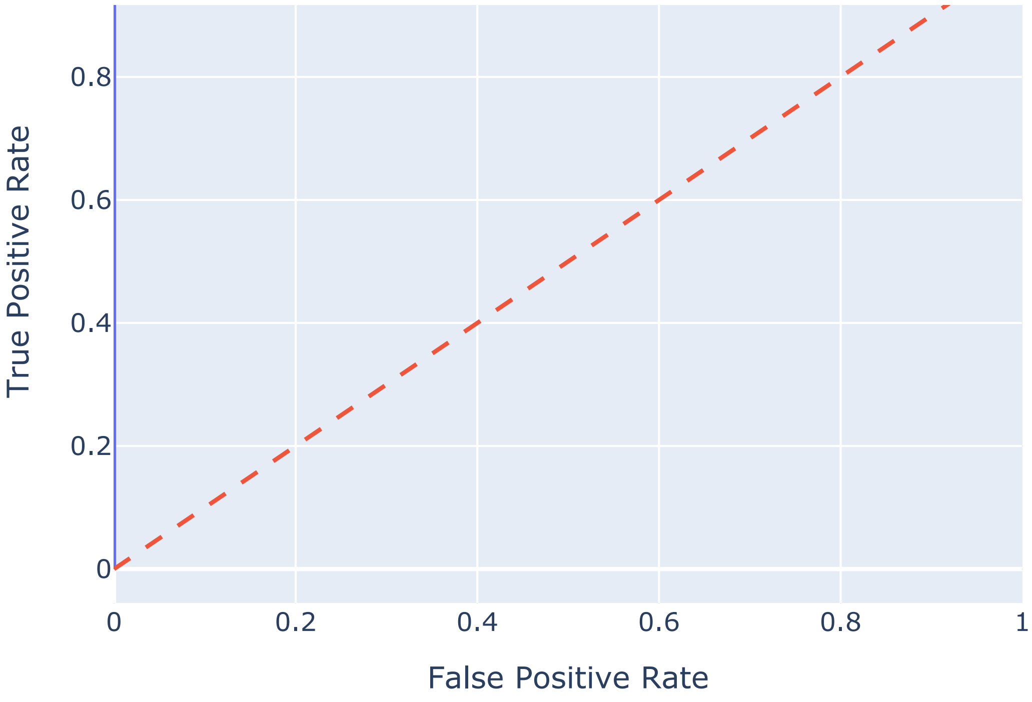


```
🔍 Target: Venom_Allergy | Model: RandomForest
📈 Accuracy: 0.9792
🎯 F1 (0): 0.9892 | F1 (1): 0.7224
📊 Precision: 0.9783 | AUC: 0.9542713864306785
📊 Confusion Matrix:
 [[2254    0]
 [   0   97]]
```

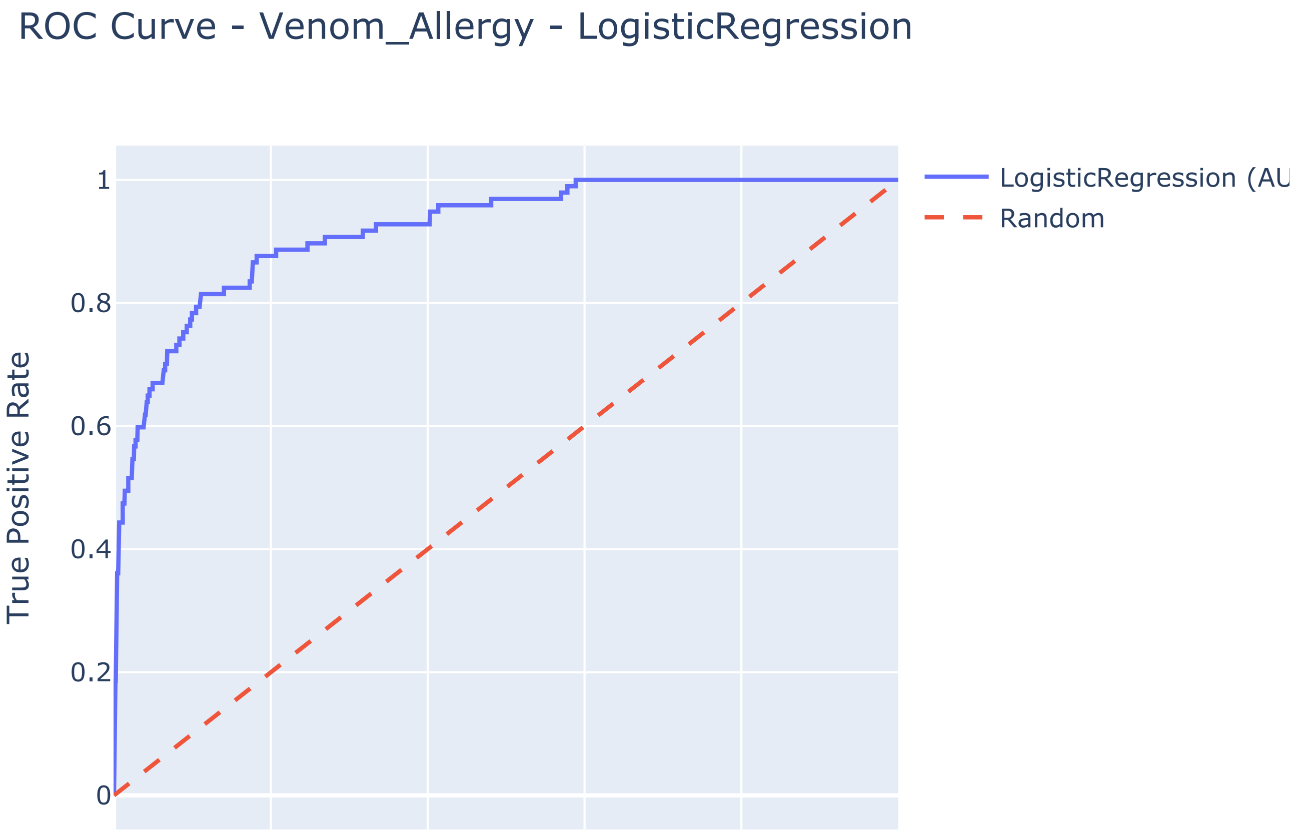

ROC Curve - Venom_Allergy - RandomForest

```
🔍 Target: Venom_Allergy | Model: XGBoost
📈 Accuracy: 0.9821
🎯 F1 (0): 0.9907 | F1 (1): 0.7764
📊 Precision: 0.9819 | AUC: 0.9512725663716814
📊 Confusion Matrix:
 [[2254    0]
 [   0   97]]
```

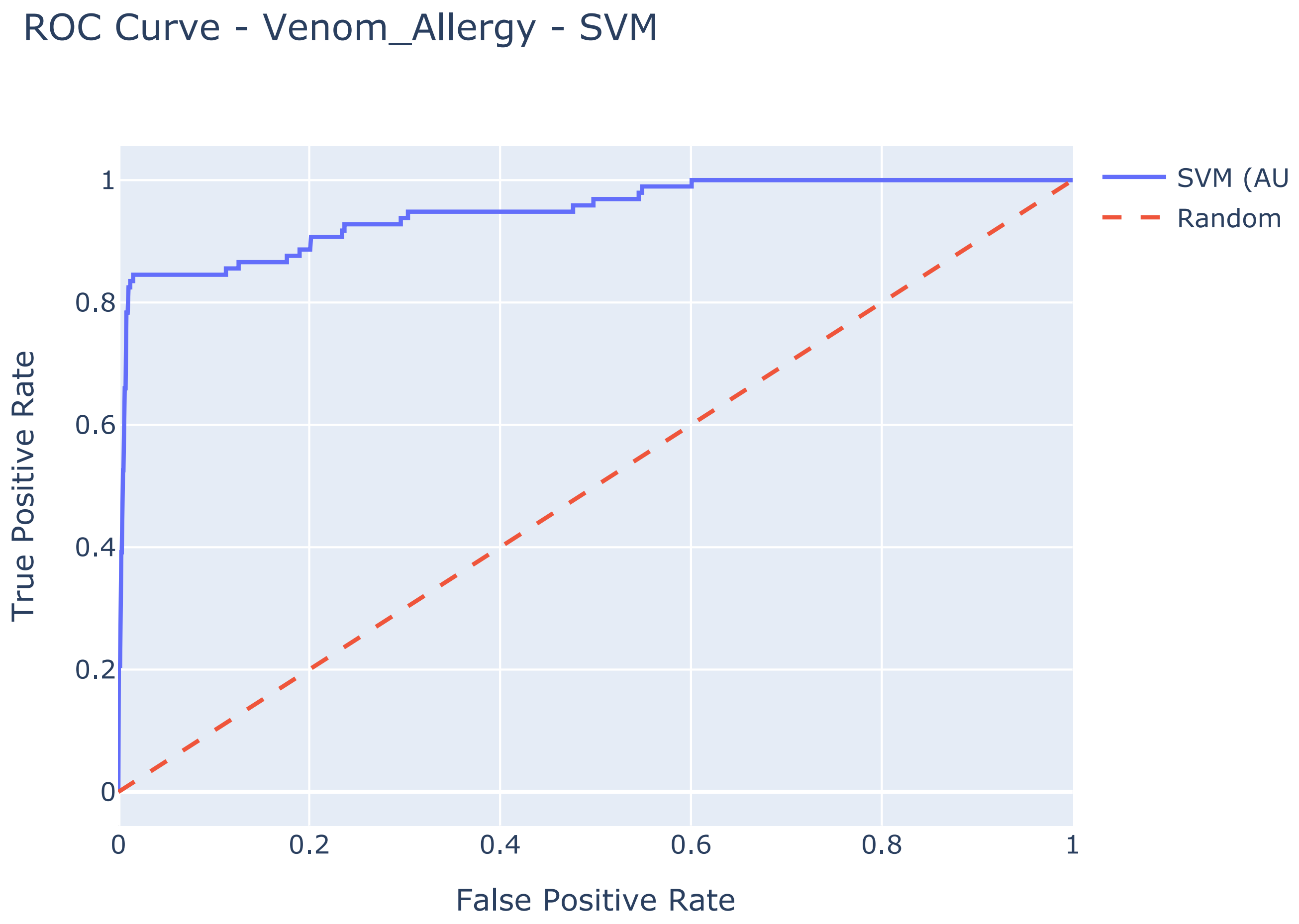

ROC Curve - Venom_Allergy - XGBoost



```
🔍 Target: Venom_Allergy | Model: LogisticRegression
📈 Accuracy: 0.9145
🎯 F1 (0): 0.9543 | F1 (1): 0.3347
📊 Precision: 0.9488 | AUC: 0.8160660766961652
📊 Confusion Matrix:
 [[2249    5]
 [  71   26]]
```



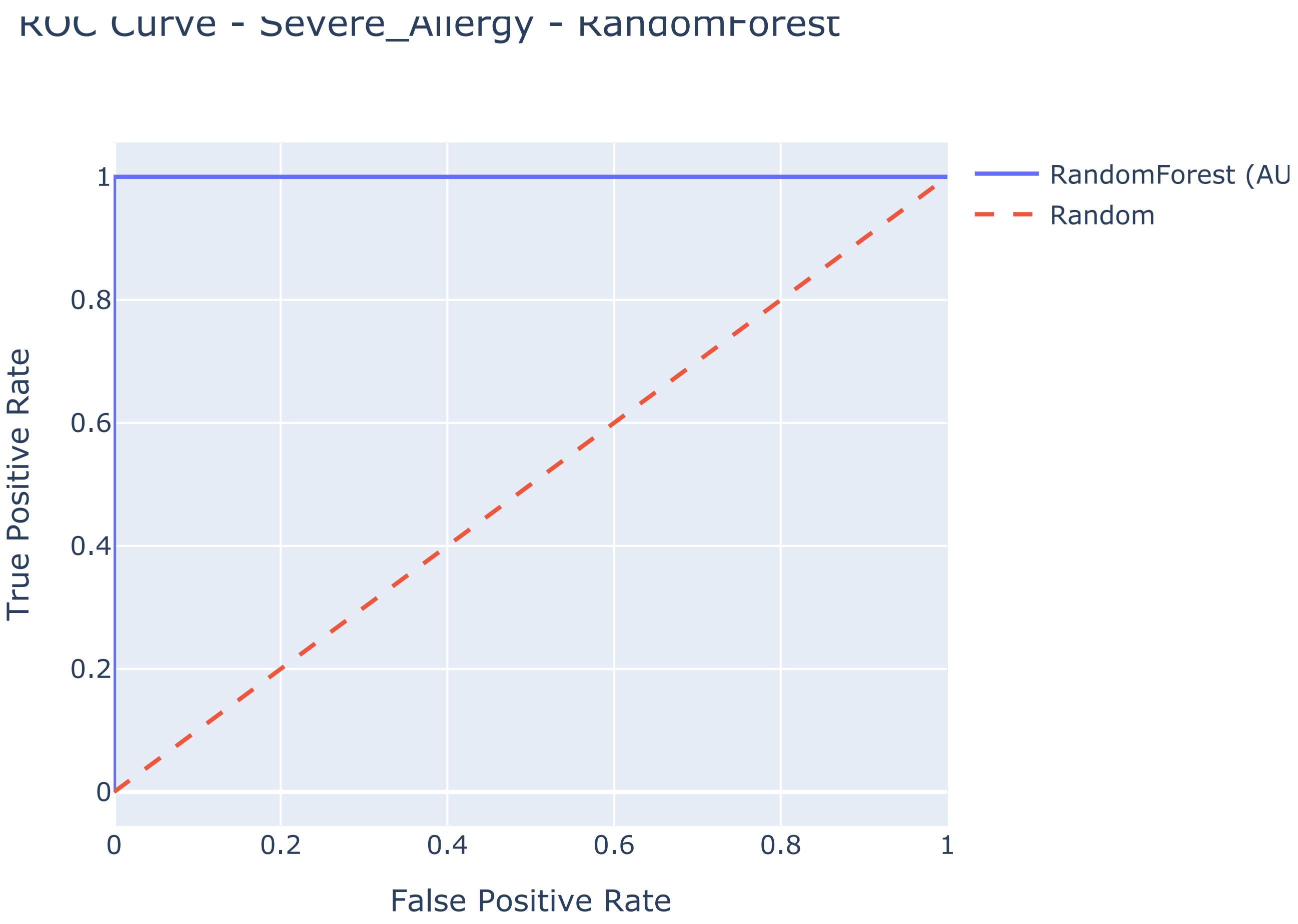
```
🔍 Target: Venom_Allergy | Model: SVM
📈 Accuracy: 0.8252
🎯 F1 (0): 0.9015 | F1 (1): 0.2187
📊 Precision: 0.9444 | AUC: 0.7936481809242871
📊 Confusion Matrix:
 [[2254    0]
 [  97    0]]
```


ROC Curve - Venom_Allergy - SVM

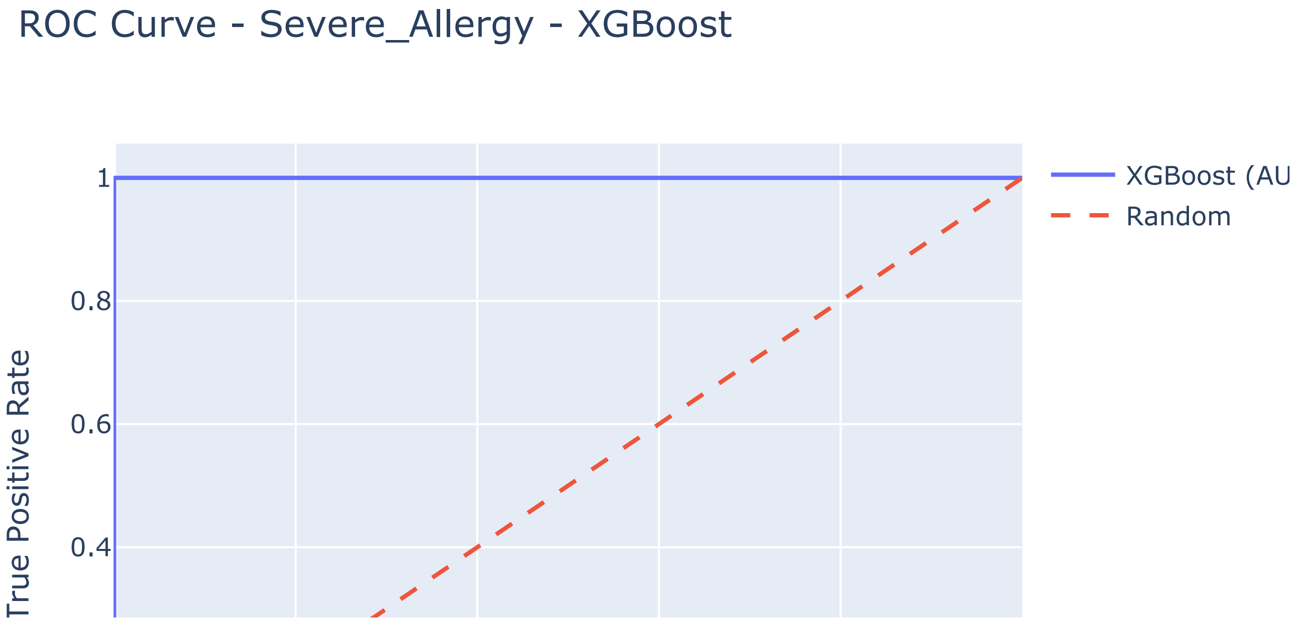
```
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedKFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.metrics import (
    f1_score, 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

V1_sev = V1[V1["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 = V1_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_severe = []
kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

for target in targets:
    y = V1_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_severe.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_severe).to_csv("results_V1_severe.csv", index=False)
```

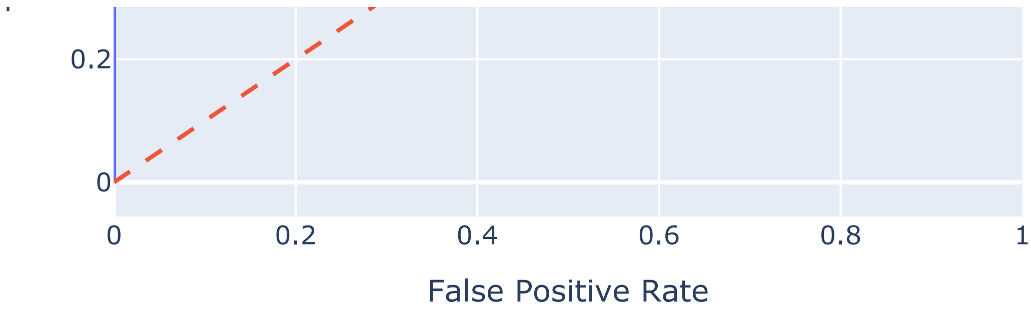
```
🔍 Target: Severe_Allergy | Model: RandomForest
📈 Accuracy: 0.8447
🎯 F1 (0): 0.7878 | F1 (1): 0.8771
📊 Precision: 0.8528 | AUC: 0.9318168604651162
📊 Confusion Matrix:
 [[430   0]
 [  0 800]]
```

ROC Curve - Severe_Allergy - RandomForest

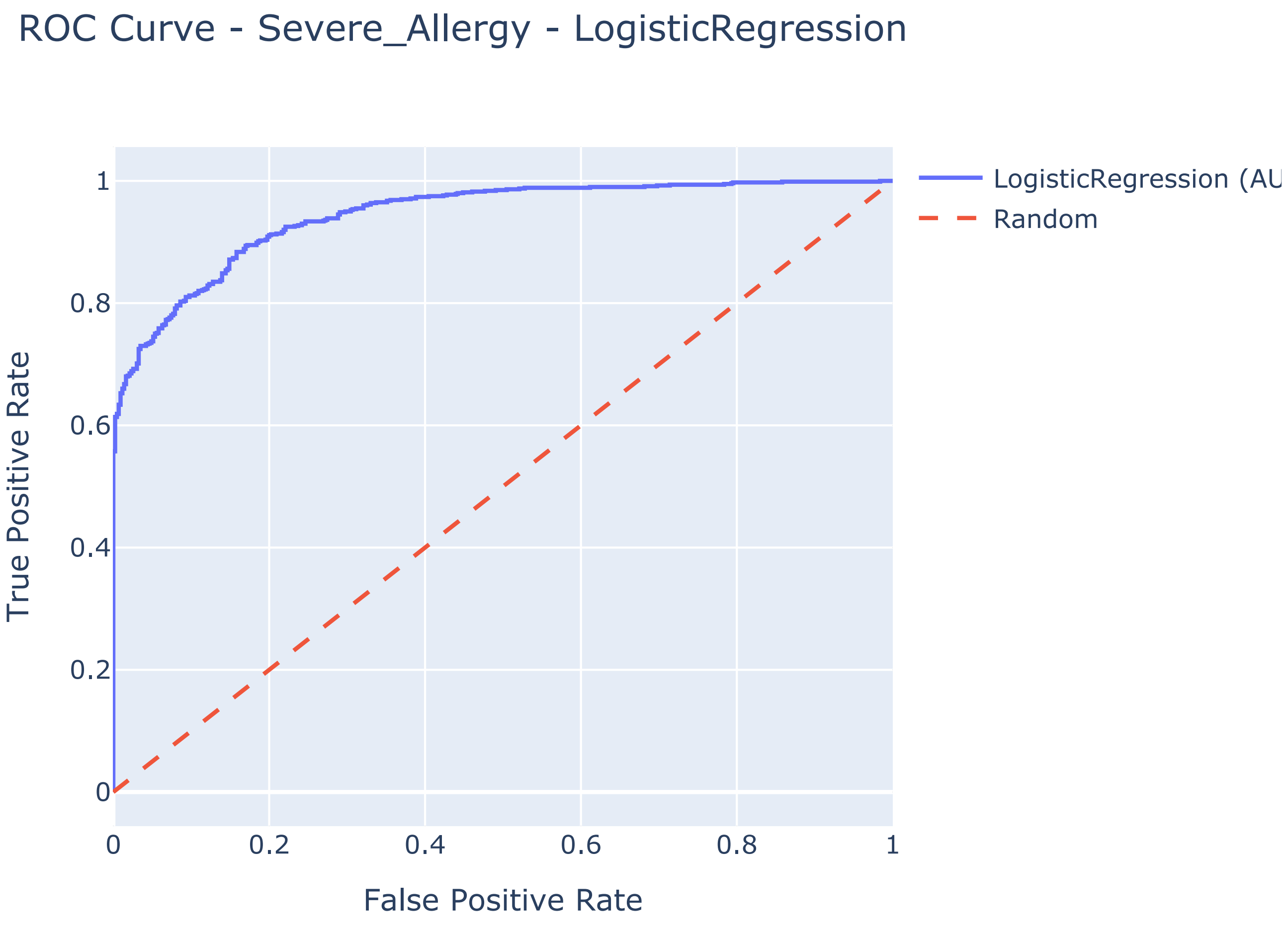


```
🔍 Target: Severe_Allergy | Model: XGBoost
📈 Accuracy: 0.8431
🎯 F1 (0): 0.7830 | F1 (1): 0.8767
📊 Precision: 0.8484 | AUC: 0.9257848837209302
📊 Confusion Matrix:
 [[430   0]
 [  0 800]]
```



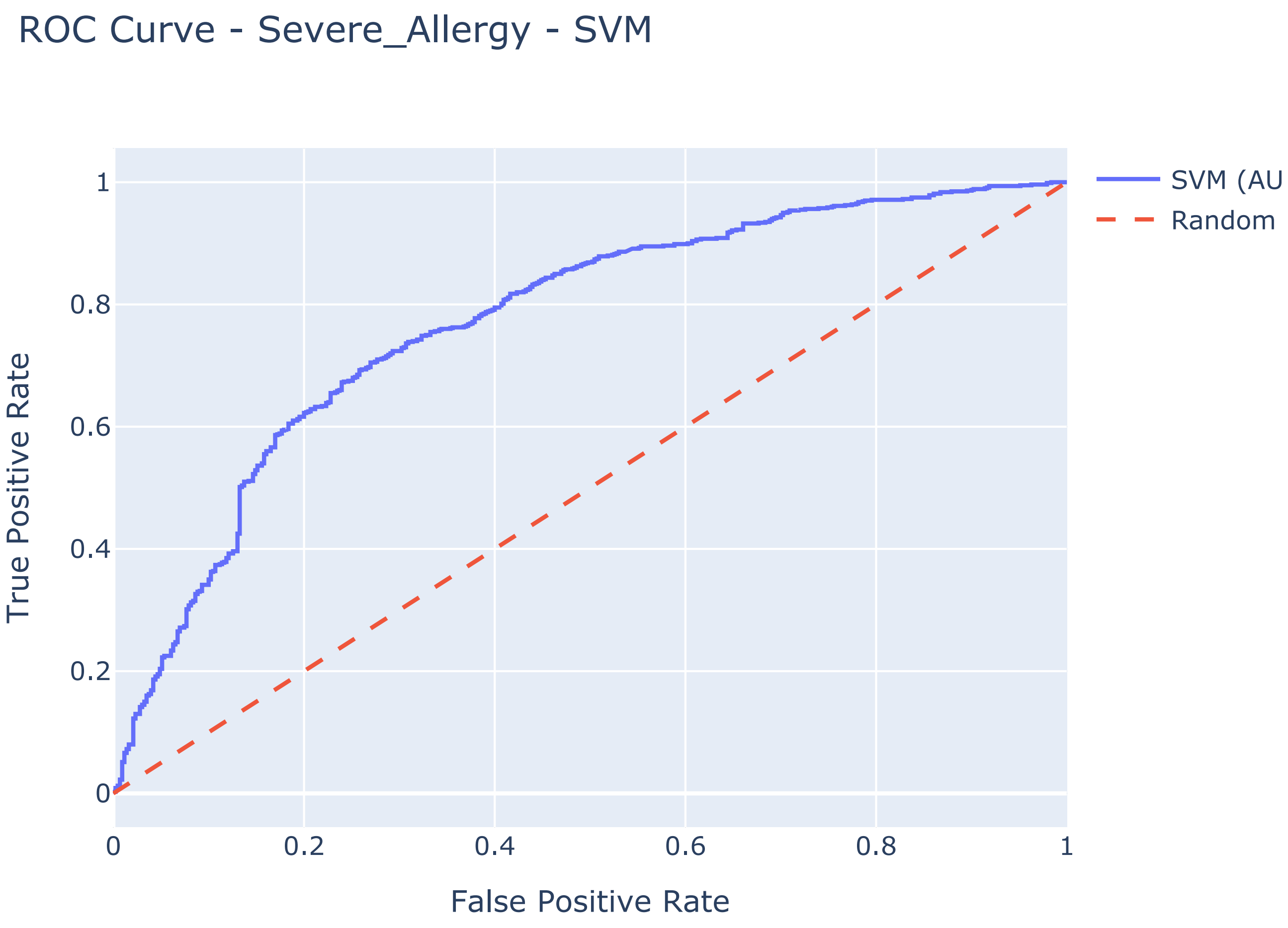


```
🔍 Target: Severe_Allergy | Model: LogisticRegression
📈 Accuracy: 0.8195
🎯 F1 (0): 0.7652 | F1 (1): 0.8527
📊 Precision: 0.8359 | AUC: 0.9182558139534883
📊 Confusion Matrix:
 [[362  68]
 [100 700]]
```



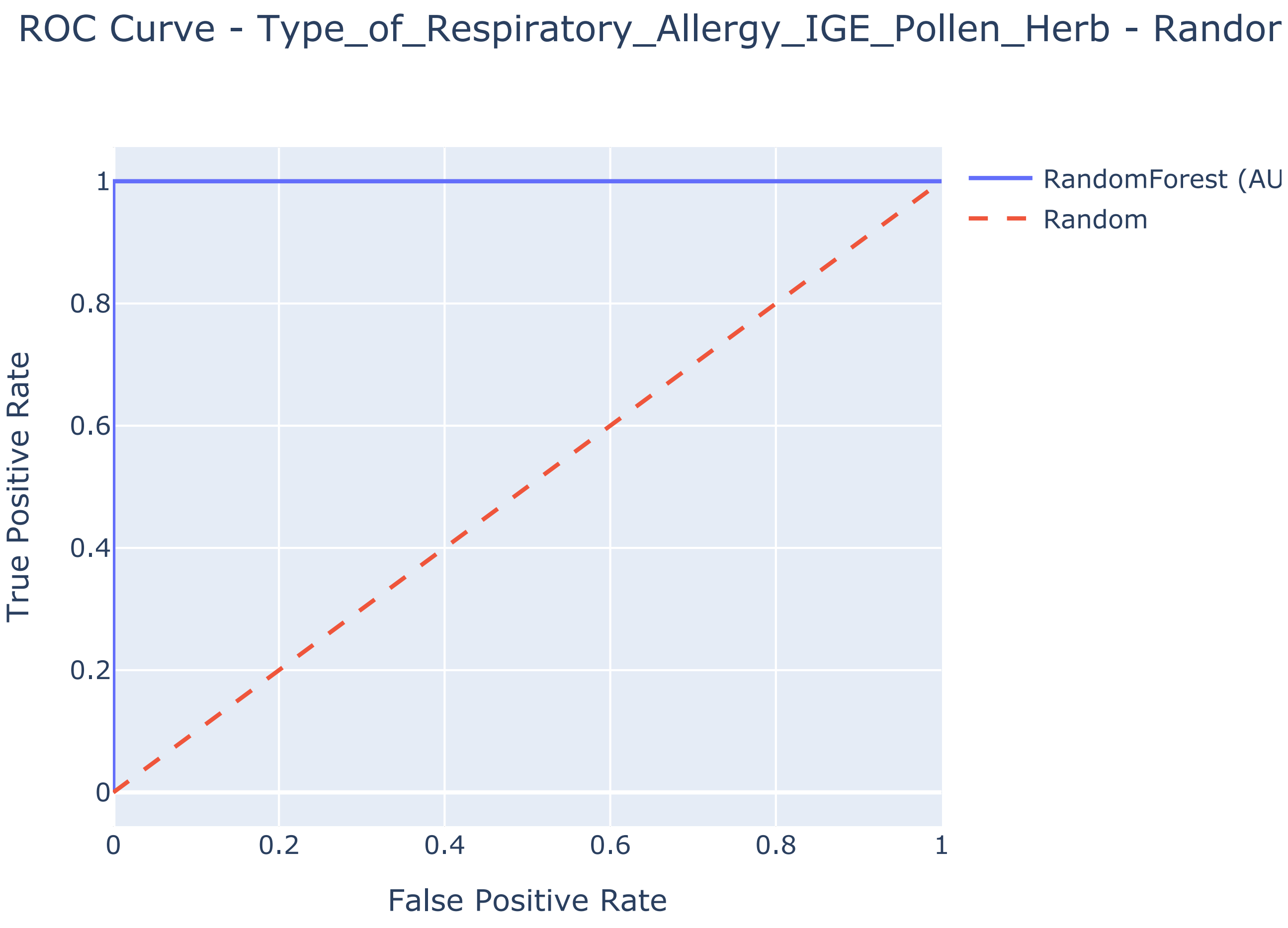
```
🔍 Target: Severe_Allergy | Model: SVM
📈 Accuracy: 0.6561
🎯 F1 (0): 0.5754 | F1 (1): 0.7102
📊 Precision: 0.6889 | AUC: 0.7212790697674418
📊 Confusion Matrix:
```

```
[[121 309]
 [ 37 763]]
```

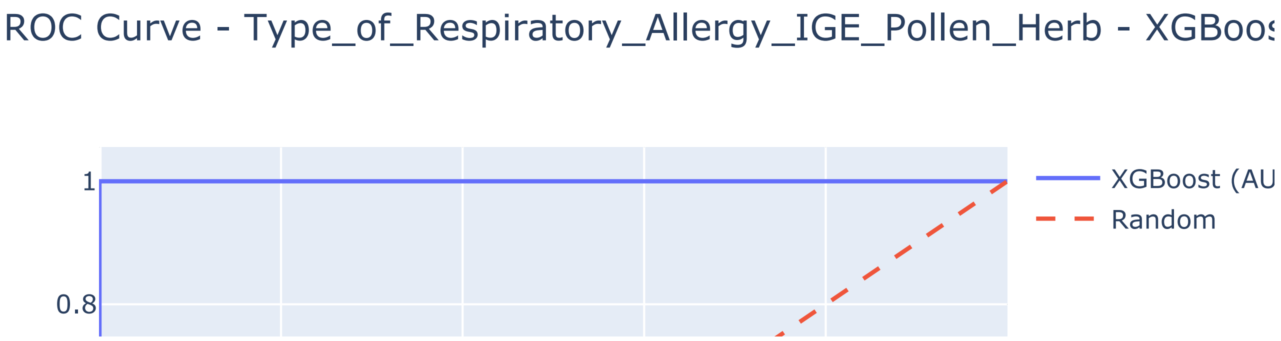


```
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedKFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.metrics import (
    f1_score, 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
V1_res = V1[V1["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 = V1_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_res = []
kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

# Boucle principale
for target in targets:
    y = V1_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_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_res).to_csv("results_V1_respiratoire.csv", index=False)
```

```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Pollen_Herb | Model: RandomFores
📈 Accuracy: 0.7413
📈 Accuracy: 0.7415
🎯 F1 (0): 0.7659 | F1 (1): 0.7081
📊 Precision: 0.7456 | AUC: 0.8438125216057385
📊 Confusion Matrix:
 [[579   0]
 [  0 484]]
```

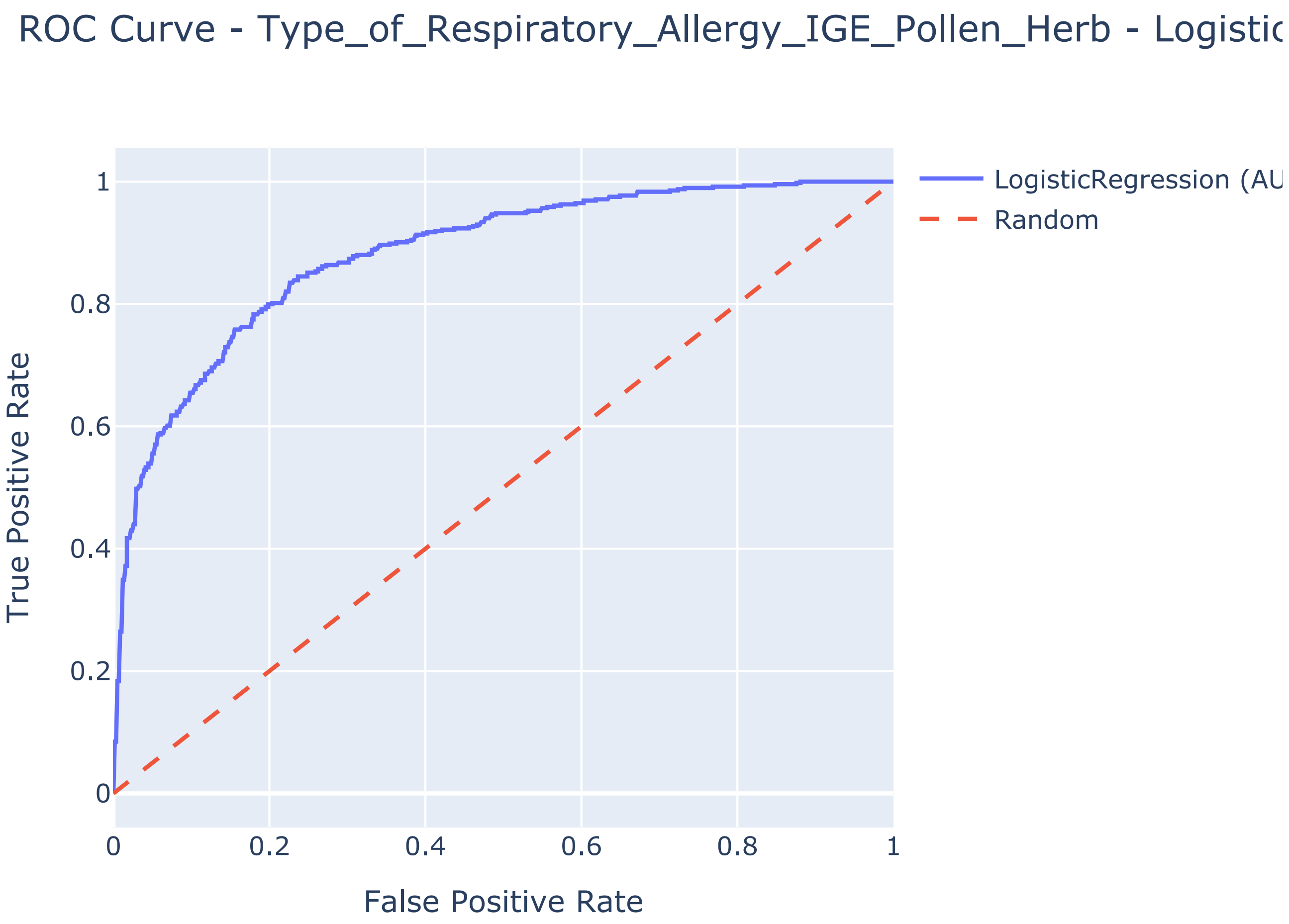


```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Pollen_Herb | Model: XGBoost
📈 Accuracy: 0.7516
🎯 F1 (0): 0.7695 | F1 (1): 0.7289
📊 Precision: 0.7556 | AUC: 0.8460580715952443
📊 Confusion Matrix:
 [[579   0]
 [  0 484]]
```

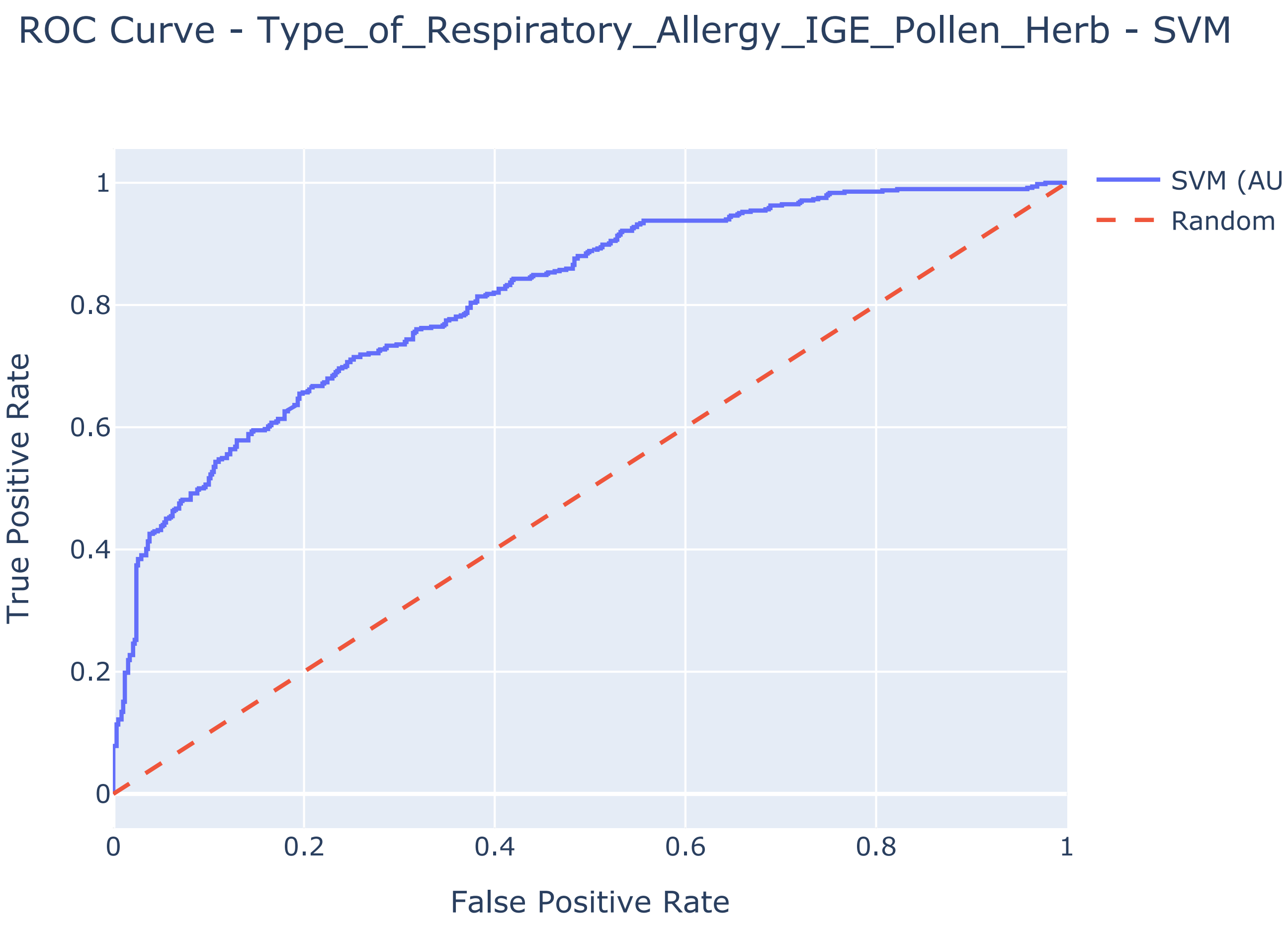


True Positive Rate
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
False Positive Rate

```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Pollen_Herb | Model: LogisticReg
📈 Accuracy: 0.7498
🎯 F1 (0): 0.7793 | F1 (1): 0.7098
📊 Precision: 0.7522 | AUC: 0.8320215949356149
📊 Confusion Matrix:
 [[509  70]
 [152 332]]
```

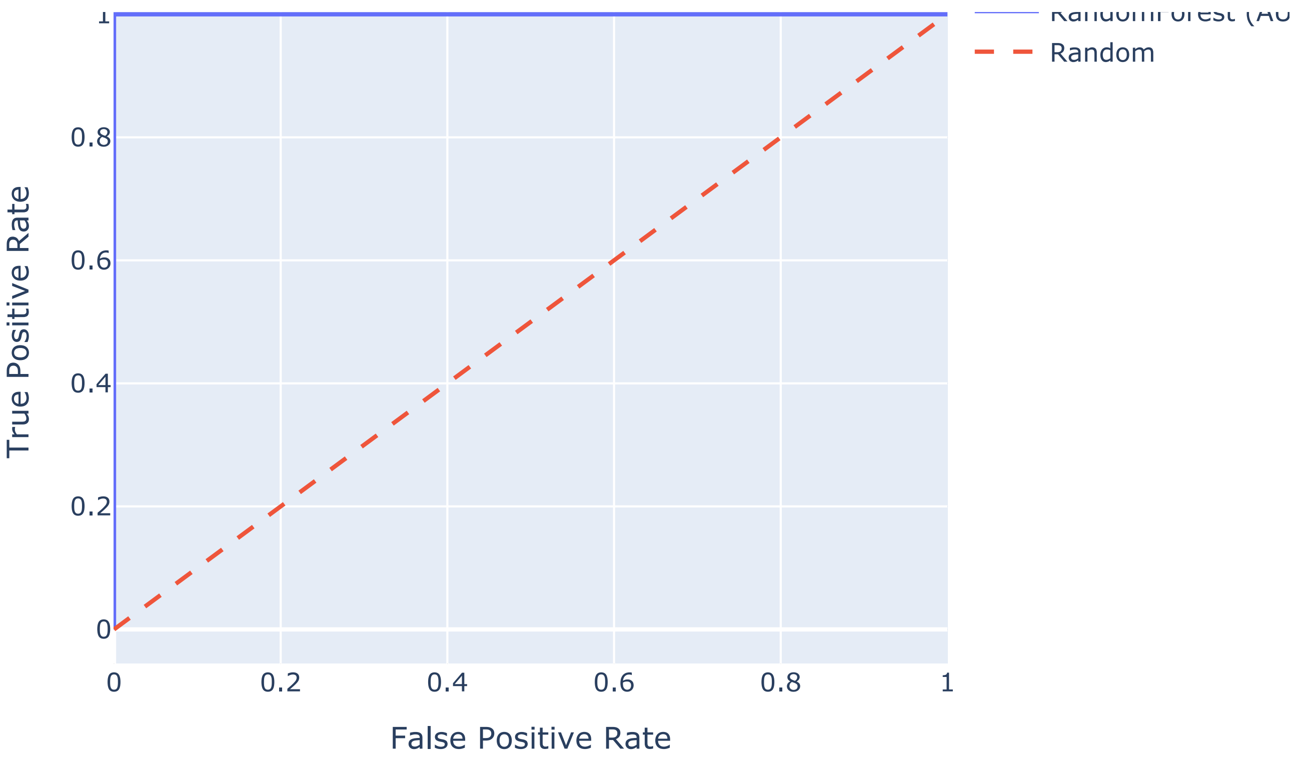

ROC Curve - Type_of_Respiratory_Allergy_IGE_Pollen_Herb - Logistic

```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Pollen_Herb | Model: SVM
📈 Accuracy: 0.7140
🎯 F1 (0): 0.7395 | F1 (1): 0.6800
📊 Precision: 0.7166 | AUC: 0.7858488585997012
📊 Confusion Matrix:
 [[475 104]
 [183 301]]
```


ROC Curve - Type_of_Respiratory_Allergy_IGE_Pollen_Herb - SVM

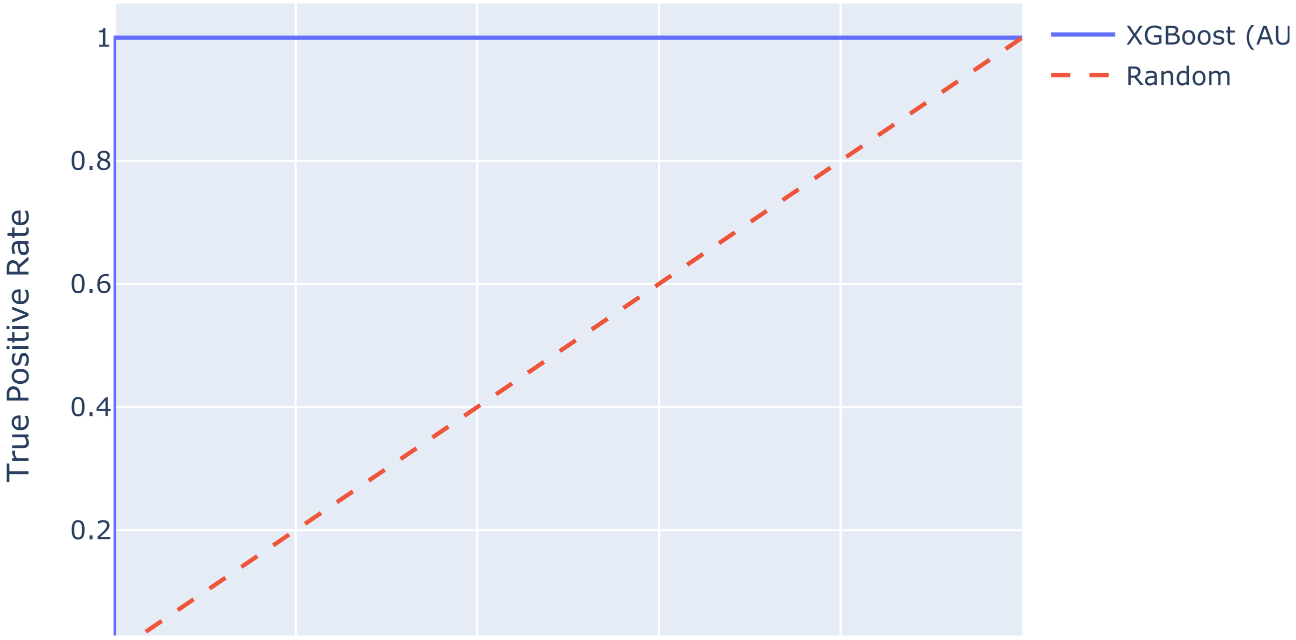
```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Pollen_Tree | Model: RandomFores
📈 Accuracy: 0.8984
🎯 F1 (0): 0.7999 | F1 (1): 0.9317
📊 Precision: 0.8980 | AUC: 0.9353242543907101
📊 Confusion Matrix:
 [[279   0]
 [  0 784]]
```

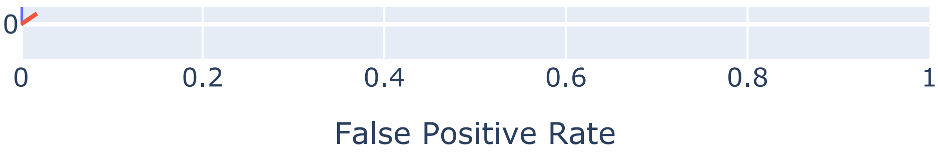
ROC Curve - Type_of_Respiratory_Allergy_IGE_Pollen_Tree - Randon



```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Pollen_Tree | Model: XGBoost
📈 Accuracy: 0.8984
🎯 F1 (0): 0.7975 | F1 (1): 0.9321
📊 Precision: 0.8972 | AUC: 0.9320739504283807
📊 Confusion Matrix:
 [[279   0]
 [  0 784]]
```

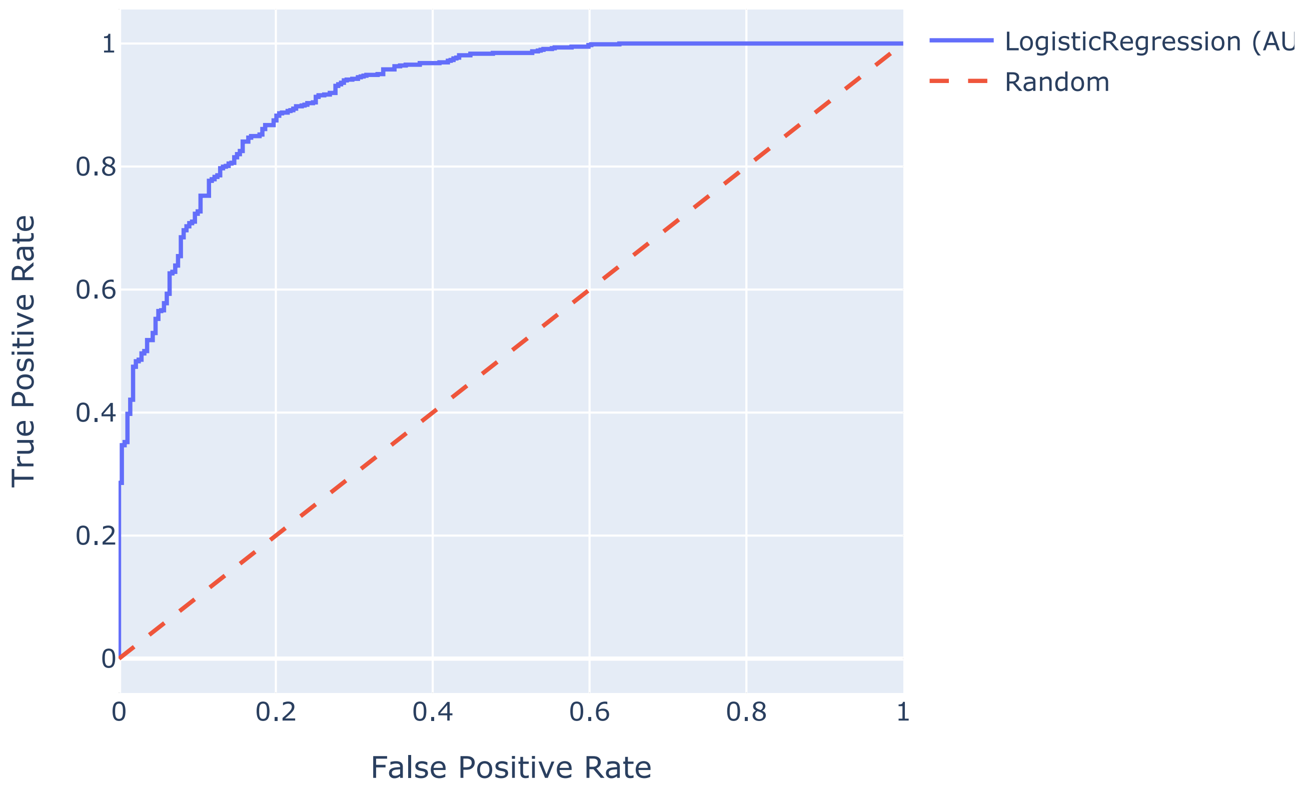
ROC Curve - Type_of_Respiratory_Allergy_IGE_Pollen_Tree - XGBoos





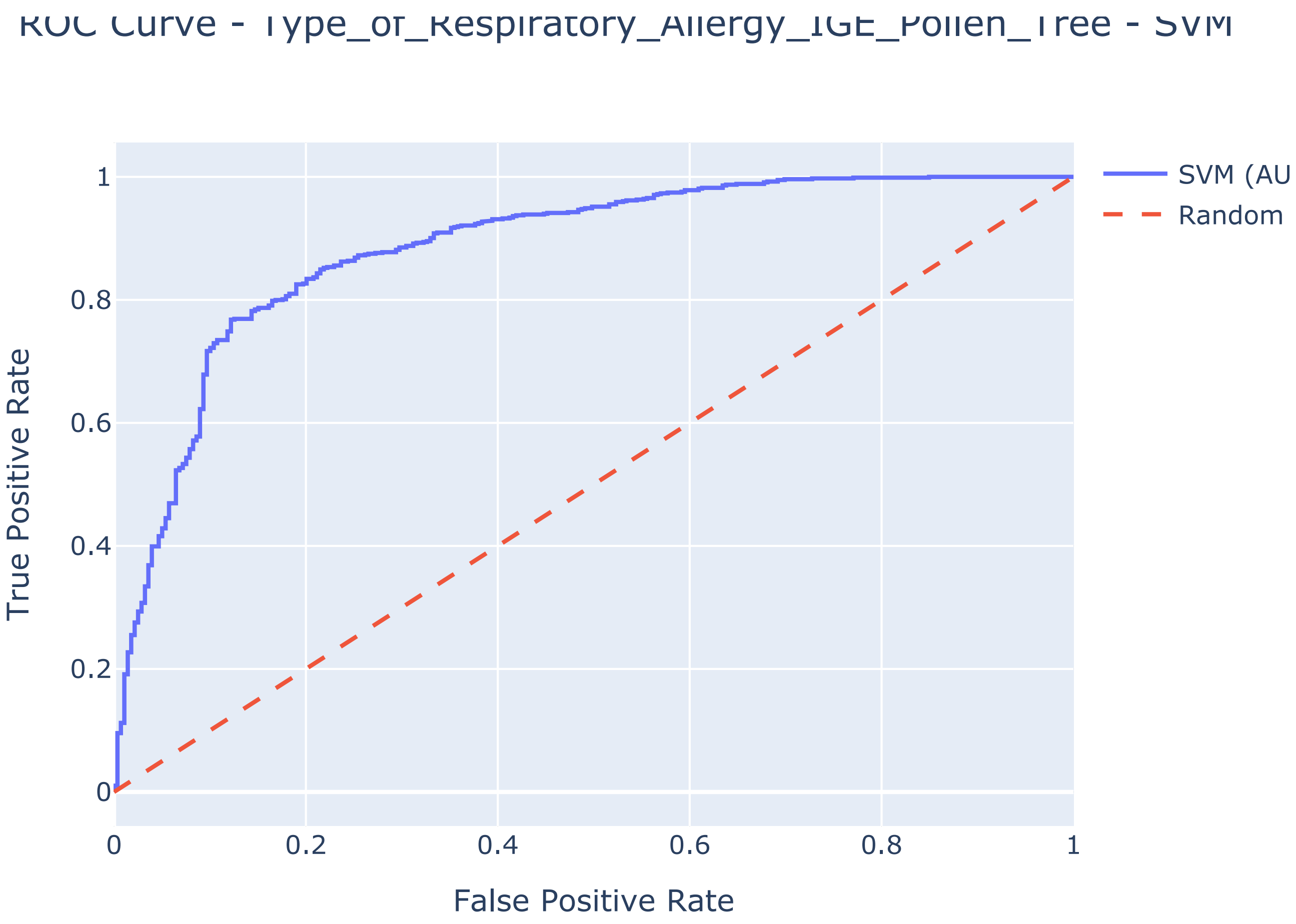
```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Pollen_Tree | Model: LogisticReg
📈 Accuracy: 0.8156
🎯 F1 (0): 0.6510 | F1 (1): 0.8745
📊 Precision: 0.8176 | AUC: 0.8584135098375605
📊 Confusion Matrix:
 [[185  94]
 [ 35 749]]
```

ROC Curve - Type_of_Respiratory_Allergy_IGE_Pollen_Tree - Logistic

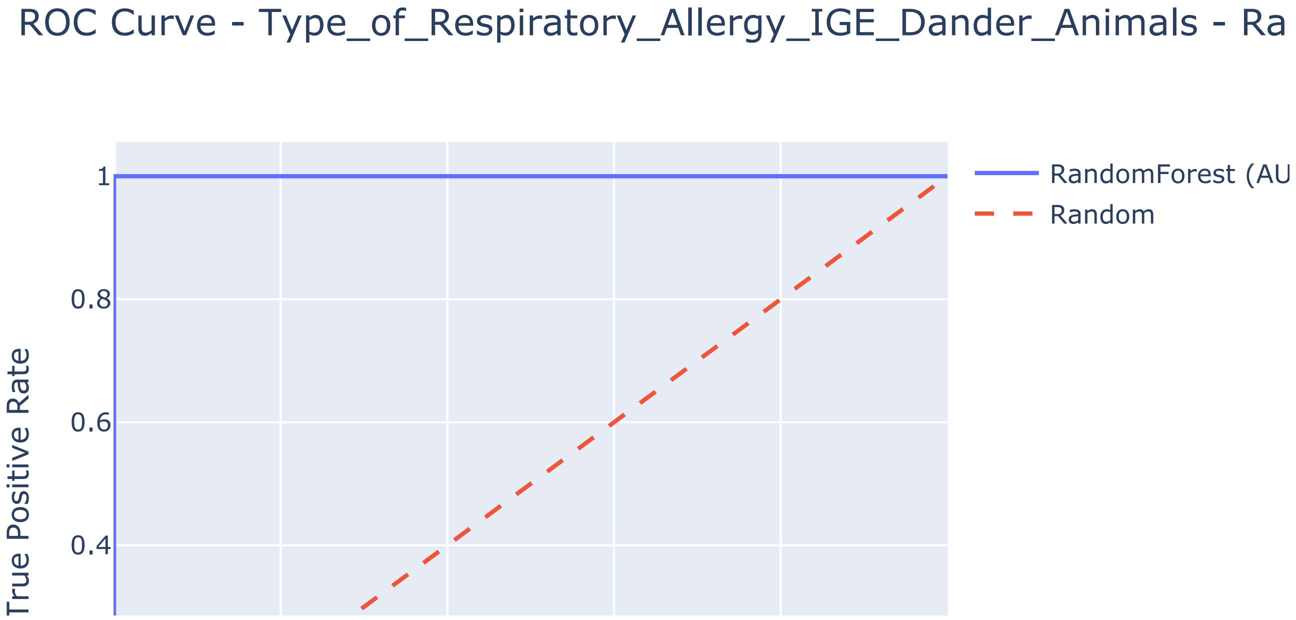


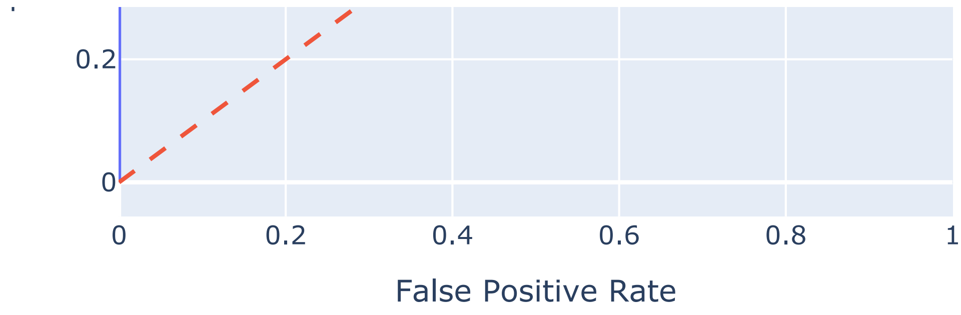
```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Pollen_Tree | Model: SVM
📈 Accuracy: 0.7761
🎯 F1 (0): 0.6663 | F1 (1): 0.8310
📊 Precision: 0.8337 | AUC: 0.8905201104568194
📊 Confusion Matrix:
 [[171 108]
 [ 57 727]]
```

ROC Curve - Type_of_Respiratory_Allergy_IGE_Pollen_Tree - SVM

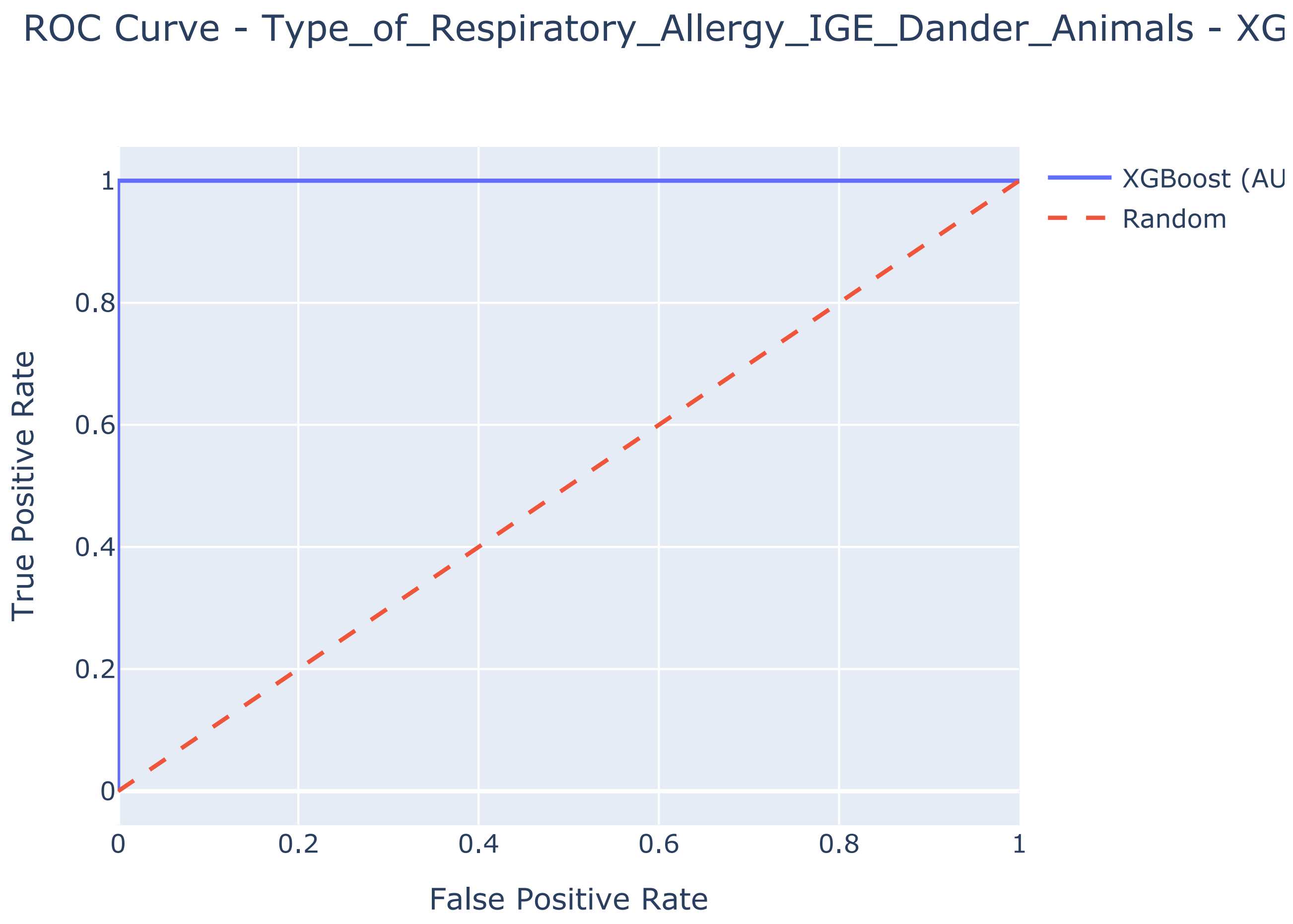


```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Dander_Animals | Model: RandomFc
📈 Accuracy: 0.8251
🎯 F1 (0): 0.7834 | F1 (1): 0.8531
📊 Precision: 0.8268 | AUC: 0.8851984834085689
📊 Confusion Matrix:
 [[458   0]
 [  0 605]]
```



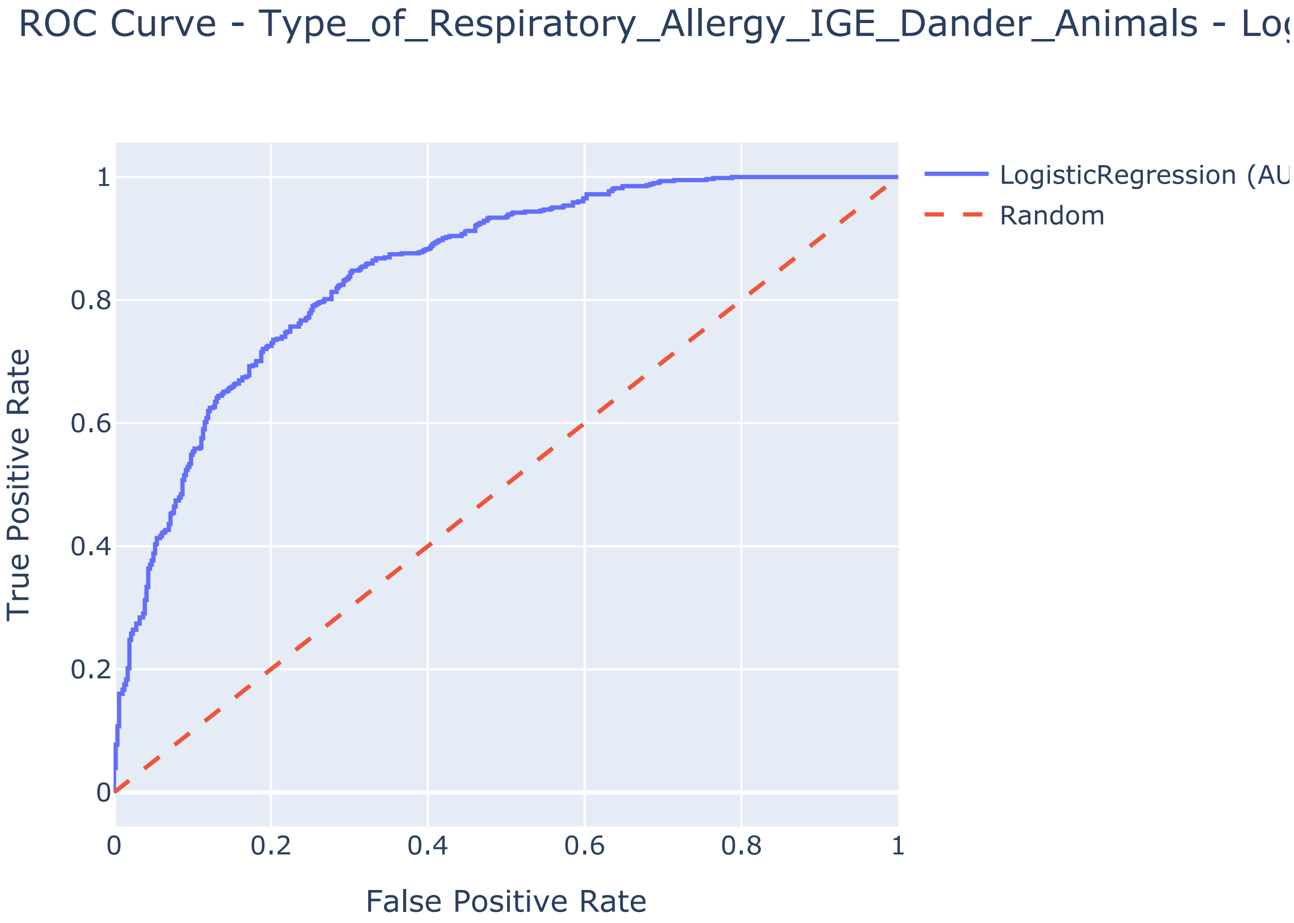


```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Dander_Animals | Model: XGBoost
📈 Accuracy: 0.8072
🎯 F1 (0): 0.7619 | F1 (1): 0.8379
📊 Precision: 0.8086 | AUC: 0.8649671339193791
📊 Confusion Matrix:
 [[458   0]
 [  0 605]]
```

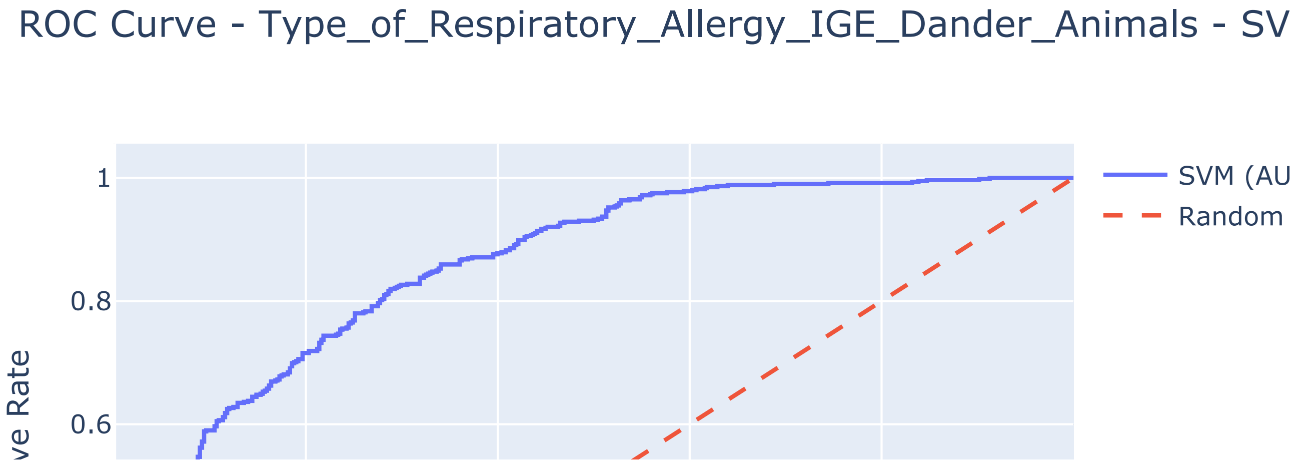


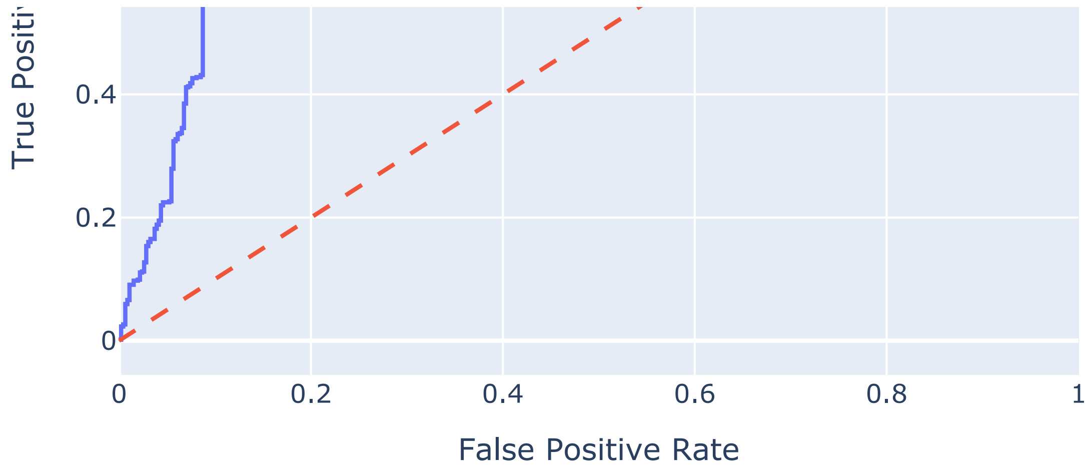
```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Dander_Animals | Model: Logistic
📈 Accuracy: 0.7169
🎯 F1 (0): 0.6911 | F1 (1): 0.7363
📊 Precision: 0.7282 | AUC: 0.7952817375465272
📊 Confusion Matrix:
```

```
[[334 124]
 [120 485]]
```

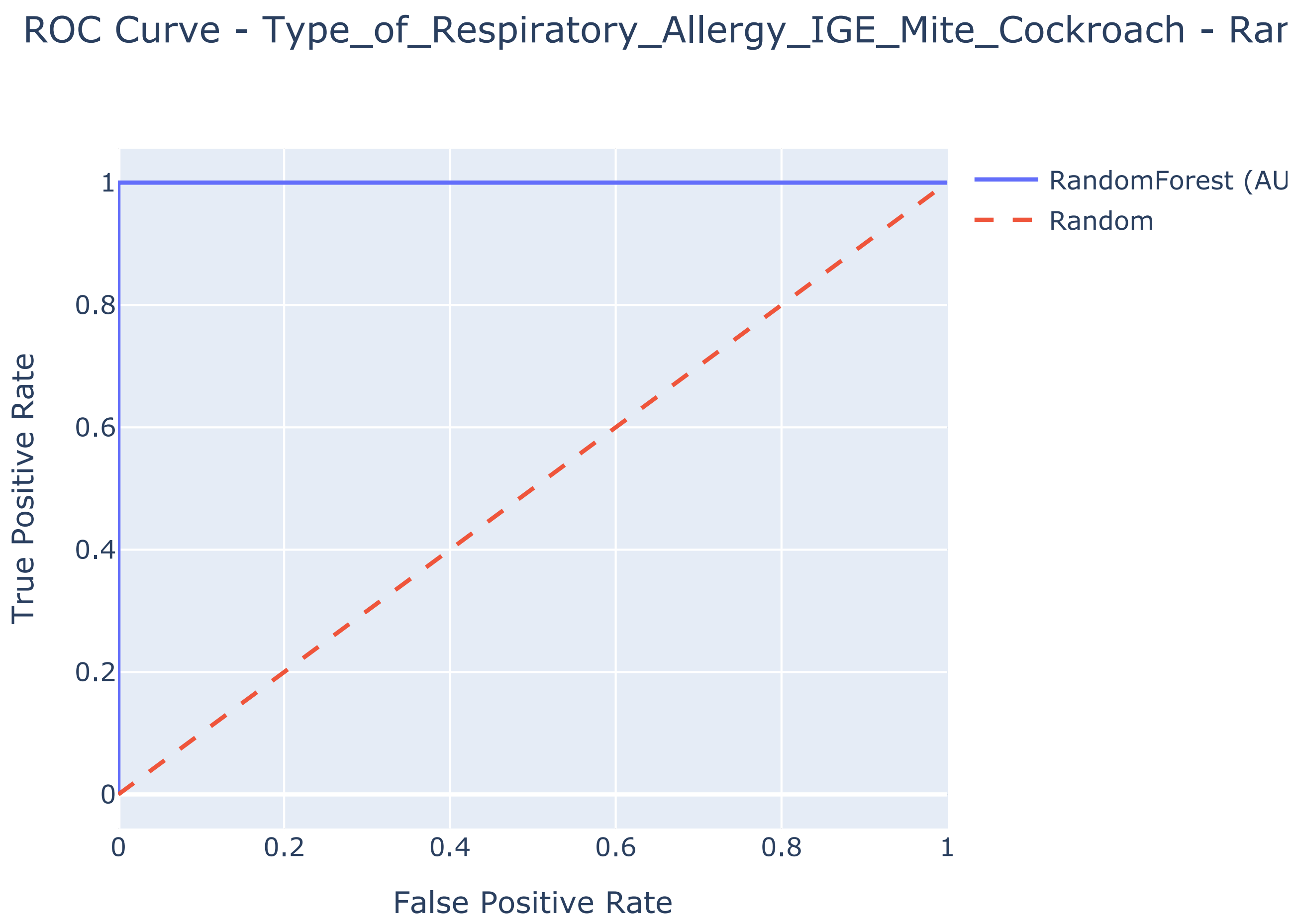


```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Dander_Animals | Model: SVM
📈 Accuracy: 0.7357
🎯 F1 (0): 0.7157 | F1 (1): 0.7523
📊 Precision: 0.7463 | AUC: 0.8254104300308862
📊 Confusion Matrix:
 [[339 119]
 [132 473]]
```



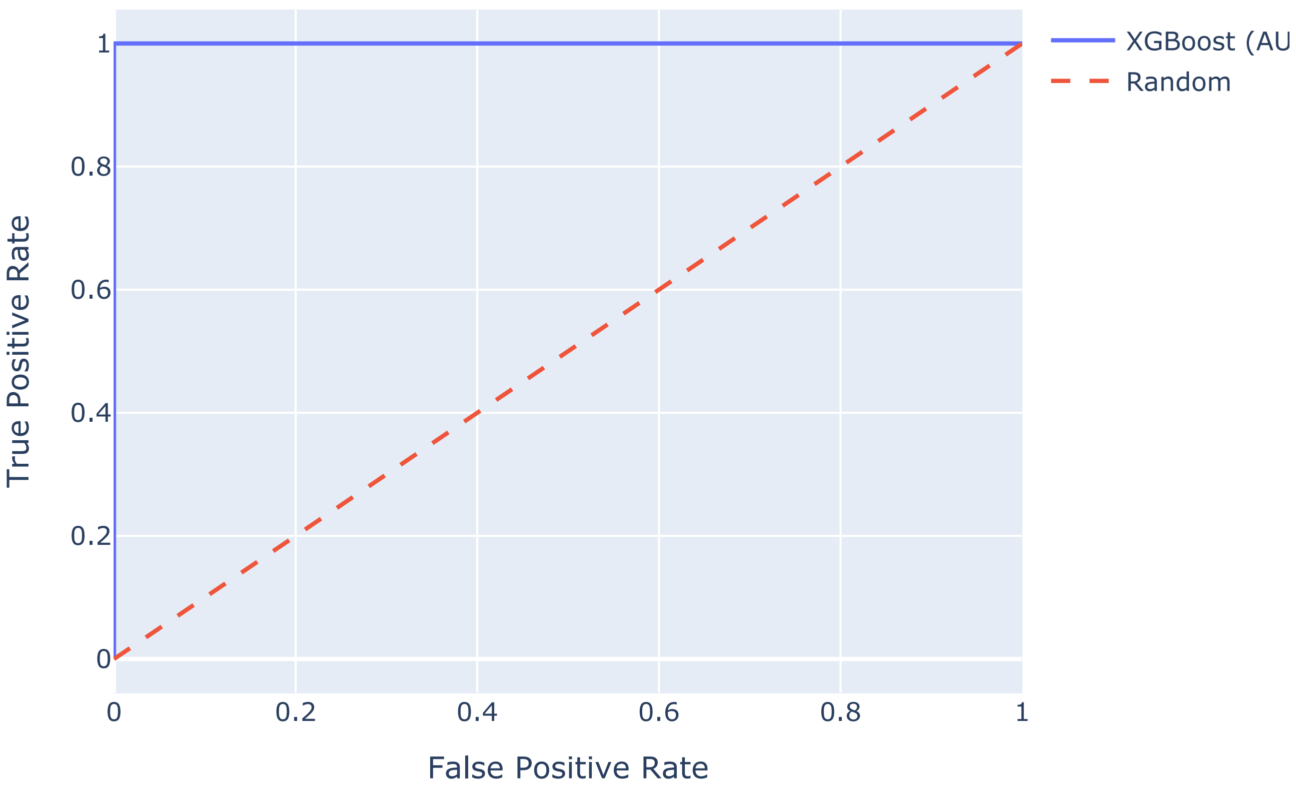


```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Mite_Cockroach | Model: RandomFc
📈 Accuracy: 0.8439
🎯 F1 (0): 0.8130 | F1 (1): 0.8656
📊 Precision: 0.8465 | AUC: 0.9245571491661309
📊 Confusion Matrix:
 [[464   0]
 [  0 599]]
```



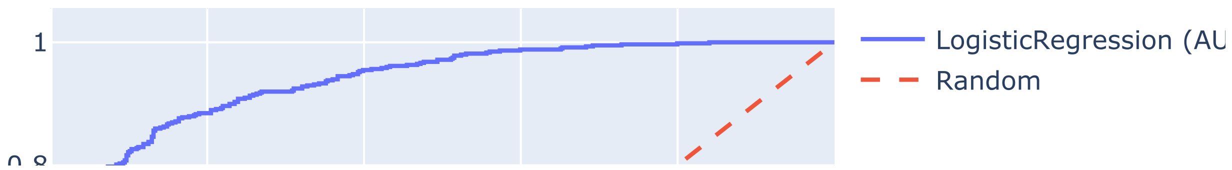
```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Mite_Cockroach | Model: XGBoost
🔍 Target: Type_of_Respiratory_Allergy_IGE_Mite_Cockroach | Model: XGBoost
📈 Accuracy: 0.8627
🎯 F1 (0): 0.8414 | F1 (1): 0.8786
📊 Precision: 0.8644 | AUC: 0.9362430946445277
📊 Confusion Matrix:
 [[464   0]
 [  0 599]]
```

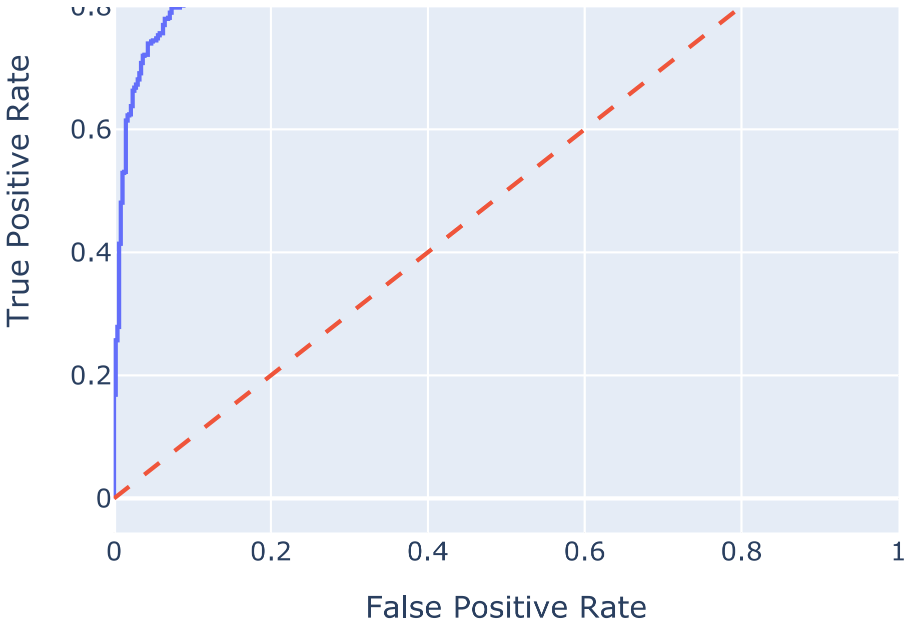
ROC Curve - Type_of_Respiratory_Allergy_IGE_Mite_Cockroach - XGI



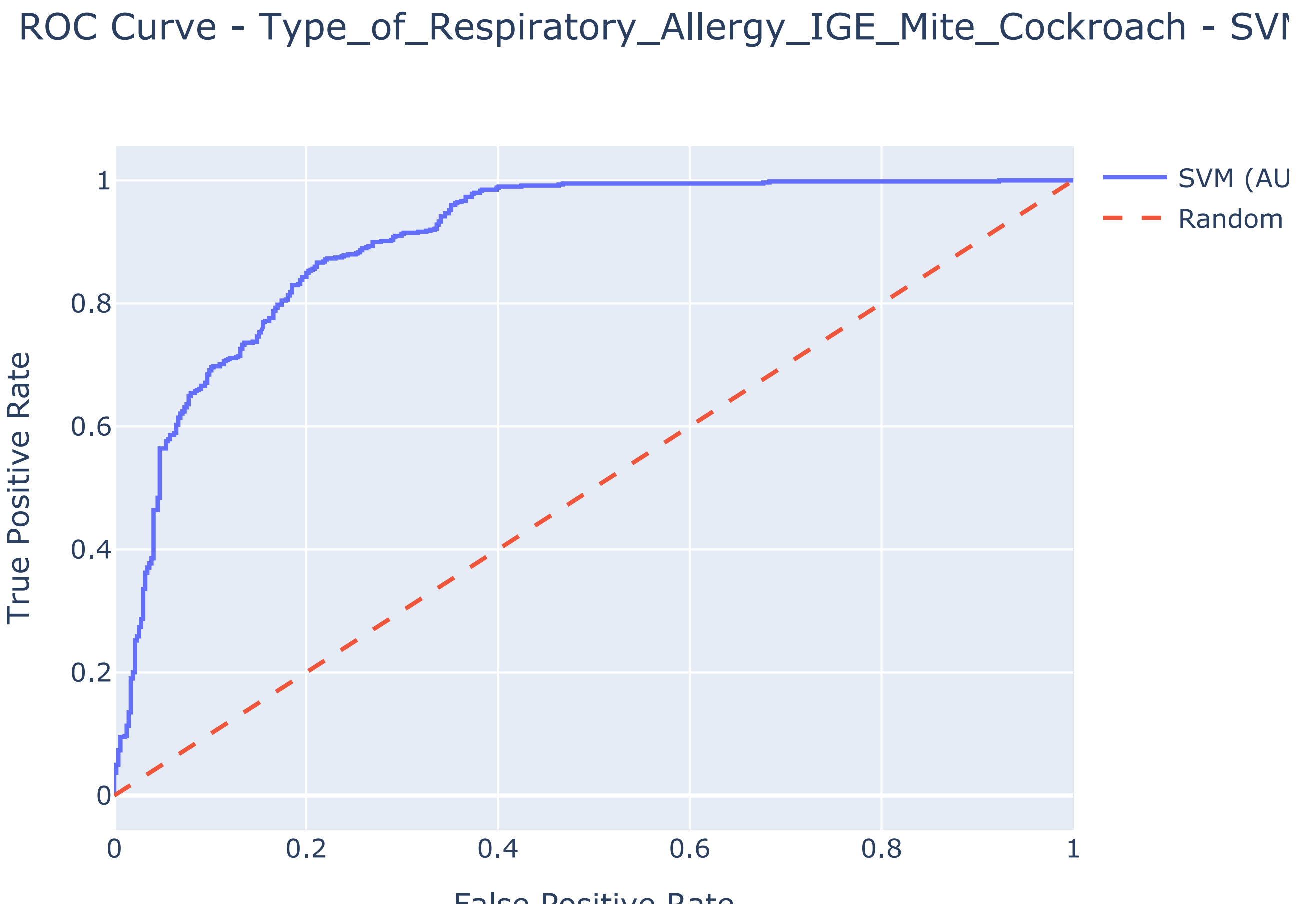
```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Mite_Cockroach | Model: Logistic
📈 Accuracy: 0.7977
🎯 F1 (0): 0.7954 | F1 (1): 0.7988
📊 Precision: 0.8222 | AUC: 0.8859456090562723
📊 Confusion Matrix:
 [[432  32]
 [132 467]]
```

ROC Curve - Type_of_Respiratory_Allergy_IGE_Mite_Cockroach - Log





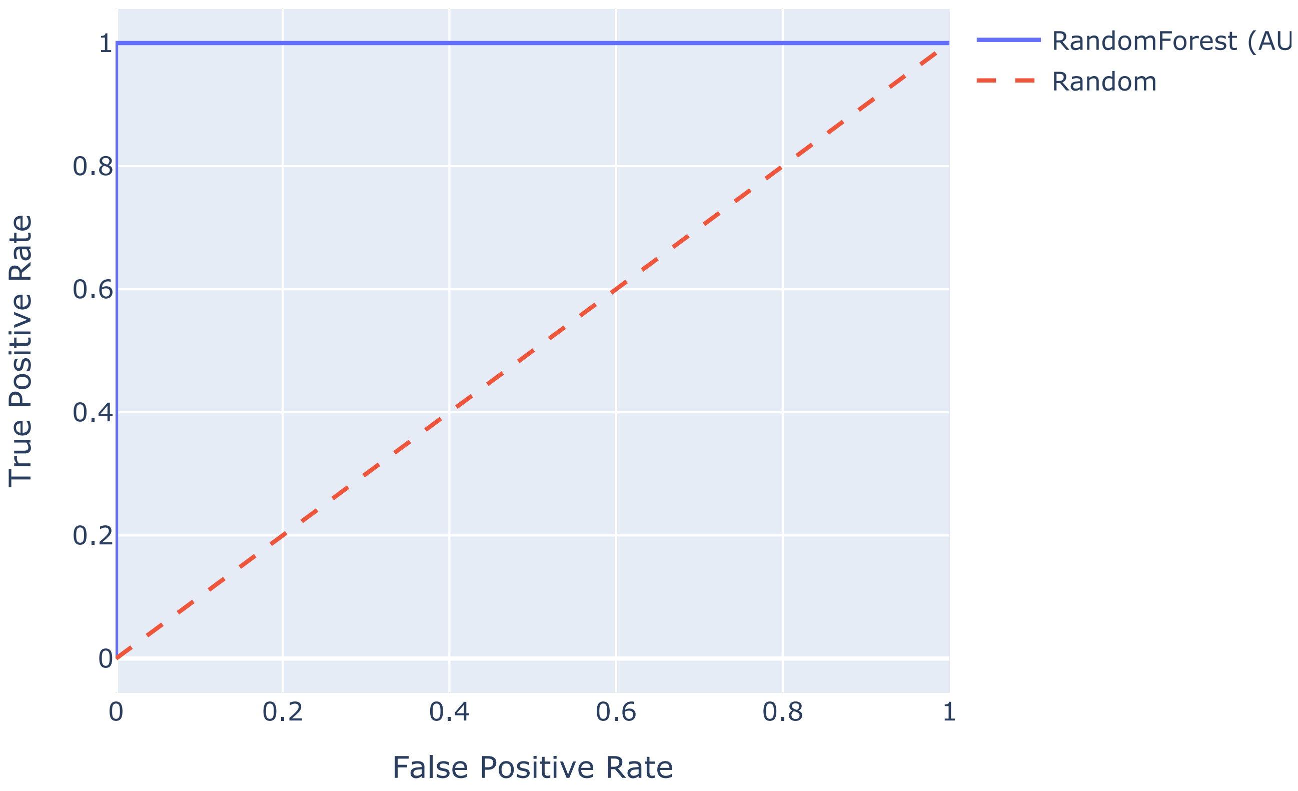
```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Mite_Cockroach | Model: SVM
📈 Accuracy: 0.7667
🎯 F1 (0): 0.7691 | F1 (1): 0.7633
📊 Precision: 0.7964 | AUC: 0.8822182196856854
📊 Confusion Matrix:
 [[413  51]
 [180 419]]
```



False Positive Rate

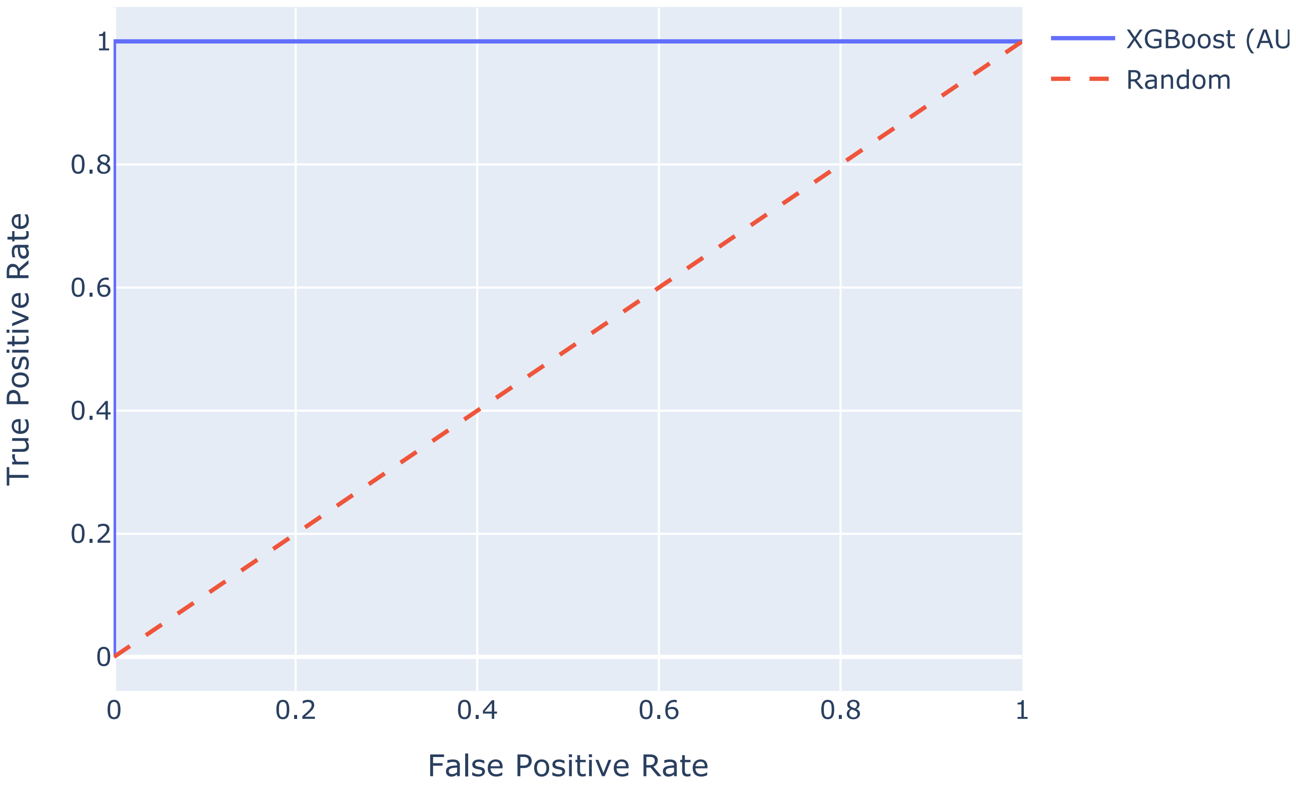
```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Molds_Yeast | Model: RandomFores
📈 Accuracy: 0.8983
🎯 F1 (0): 0.9245 | F1 (1): 0.8441
📊 Precision: 0.9092 | AUC: 0.9610034691950015
📊 Confusion Matrix:
 [[746   0]
 [  0 317]]
```

ROC Curve - Type_of_Respiratory_Allergy_IGE_Molds_Yeast - Randor

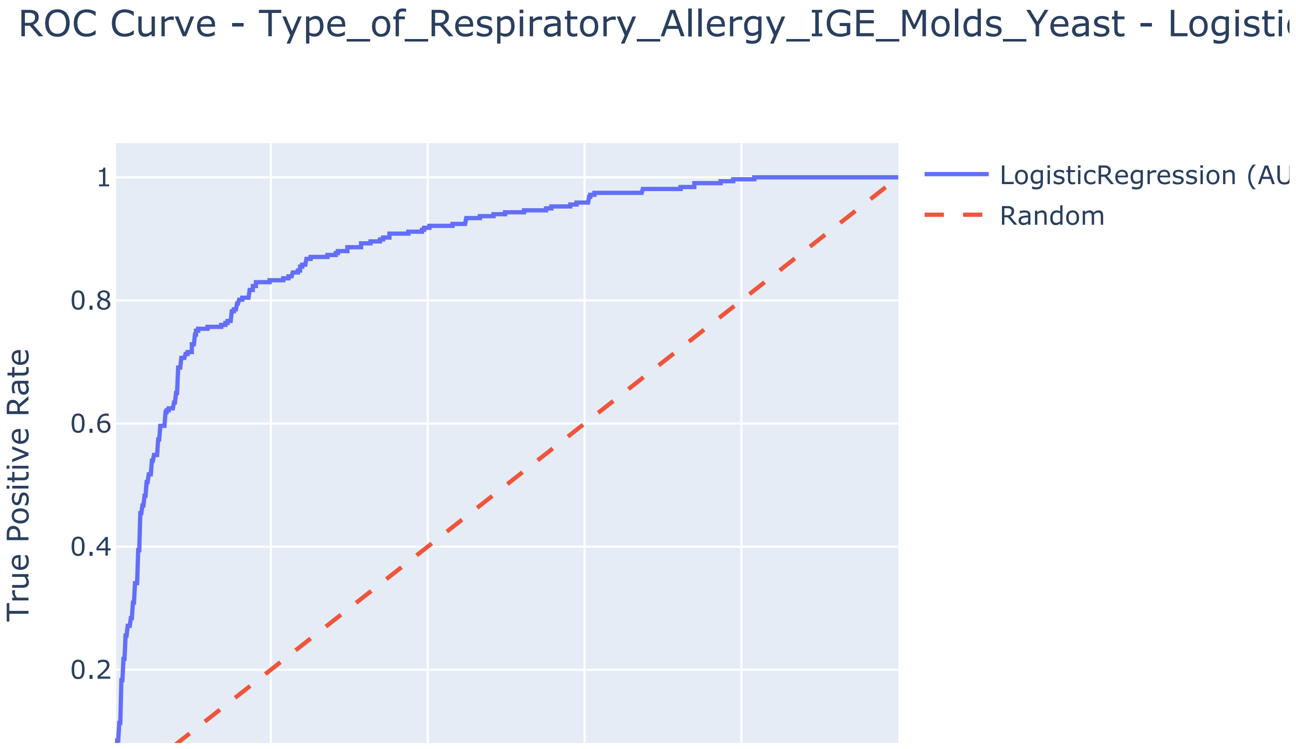


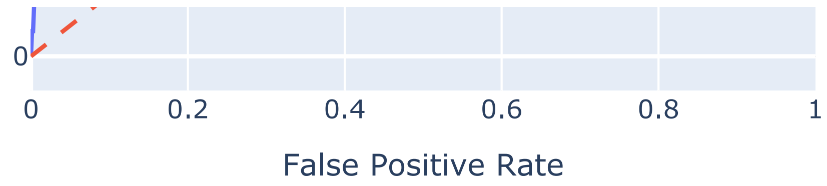
```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Molds_Yeast | Model: XGBoost
📈 Accuracy: 0.9049
🎯 F1 (0): 0.9305 | F1 (1): 0.8490
📊 Precision: 0.9113 | AUC: 0.9687050820982274
📊 Confusion Matrix:
 [[746   0]
 [  0 317]]
```

ROC Curve - Type_of_Respiratory_Allergy_IGE_Molds_Yeast - XGBoo

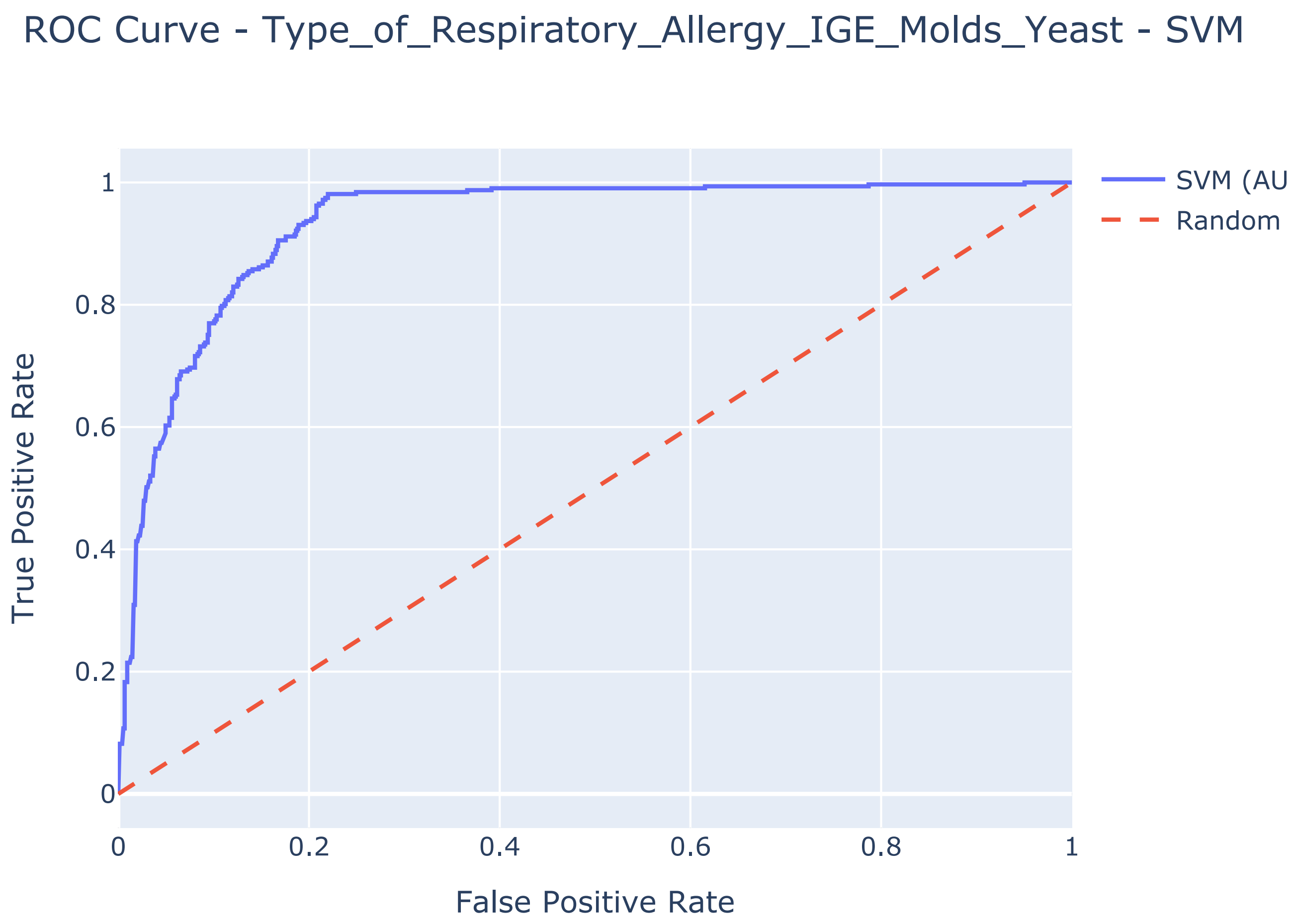


```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Molds_Yeast | Model: LogisticReg
📈 Accuracy: 0.7996
🎯 F1 (0): 0.8602 | F1 (1): 0.6443
📊 Precision: 0.7962 | AUC: 0.8110391601278698
📊 Confusion Matrix:
 [[711  35]
 [149 168]]
```

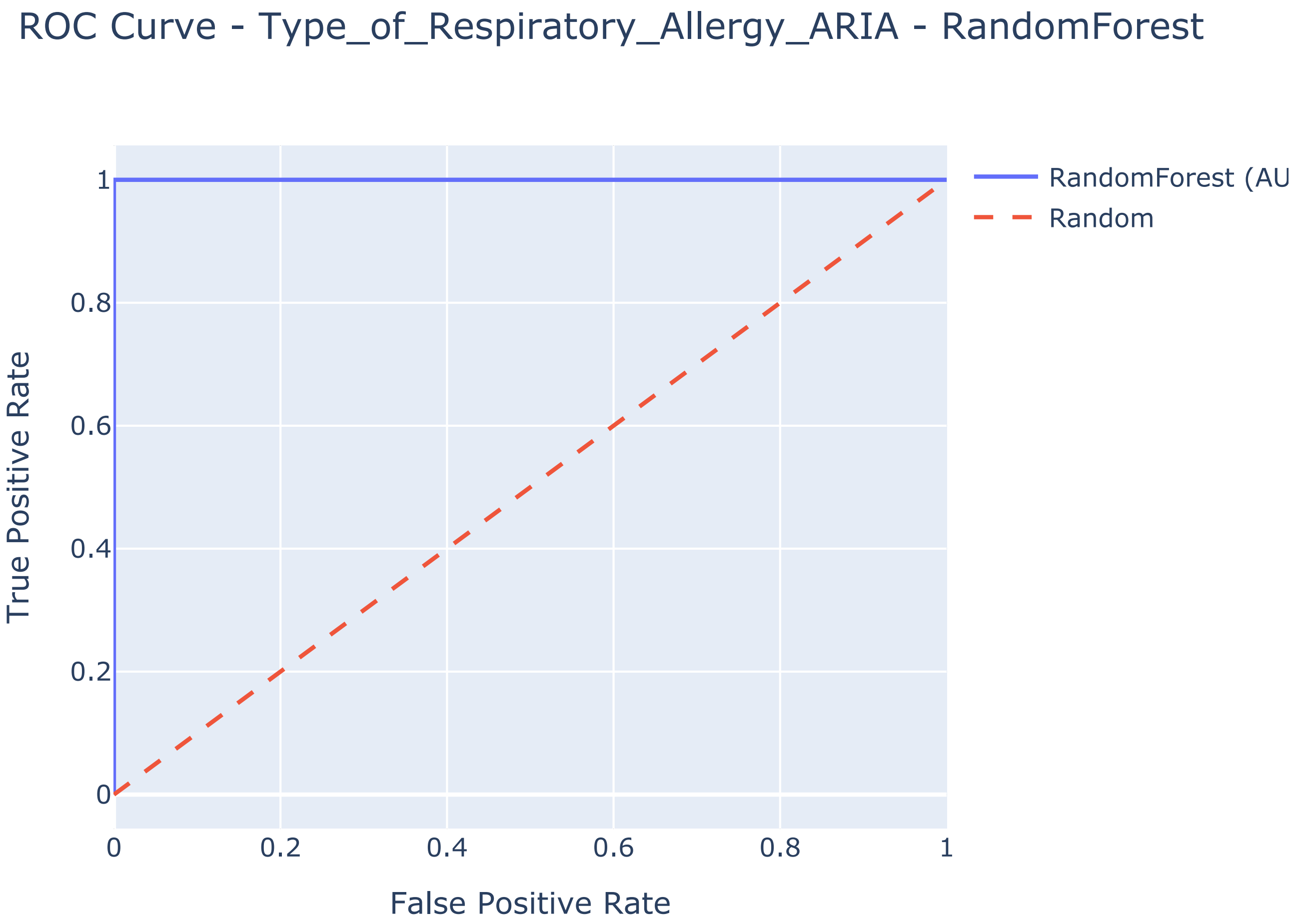

ROC Curve - Type_of_Respiratory_Allergy_IGE_Molds_Yeast - Logisti



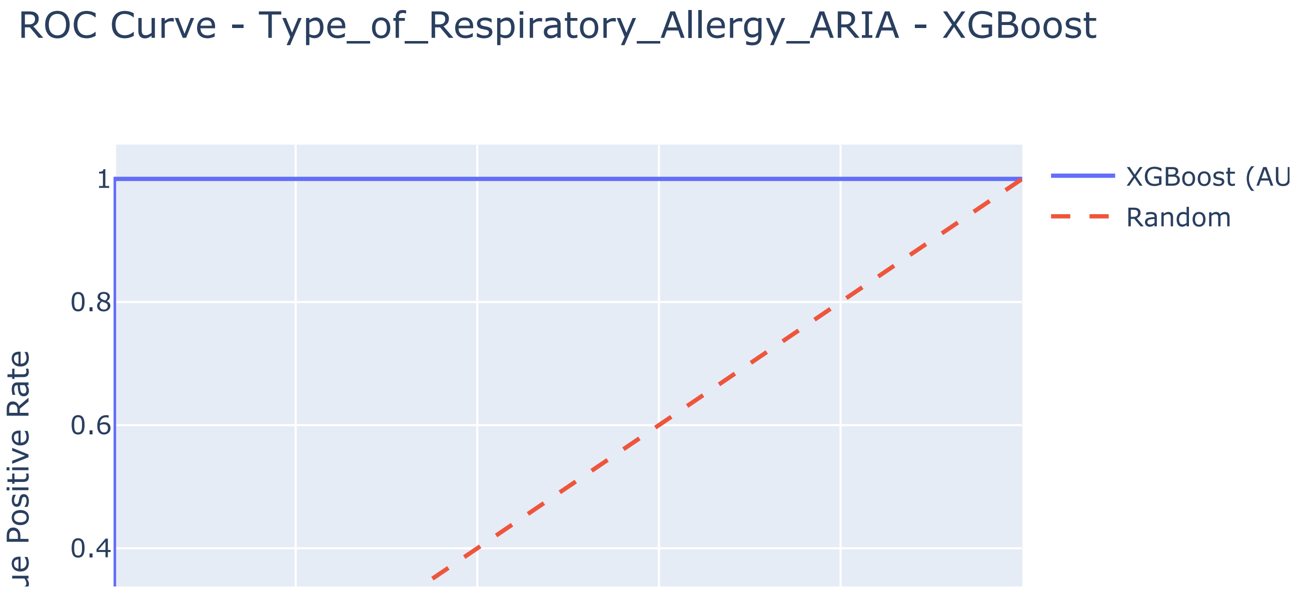
```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Molds_Yeast | Model: SVM
📈 Accuracy: 0.8230
🎯 F1 (0): 0.8734 | F1 (1): 0.7037
📊 Precision: 0.8259 | AUC: 0.8923271941296136
📊 Confusion Matrix:
 [[718  28]
 [144 173]]
```



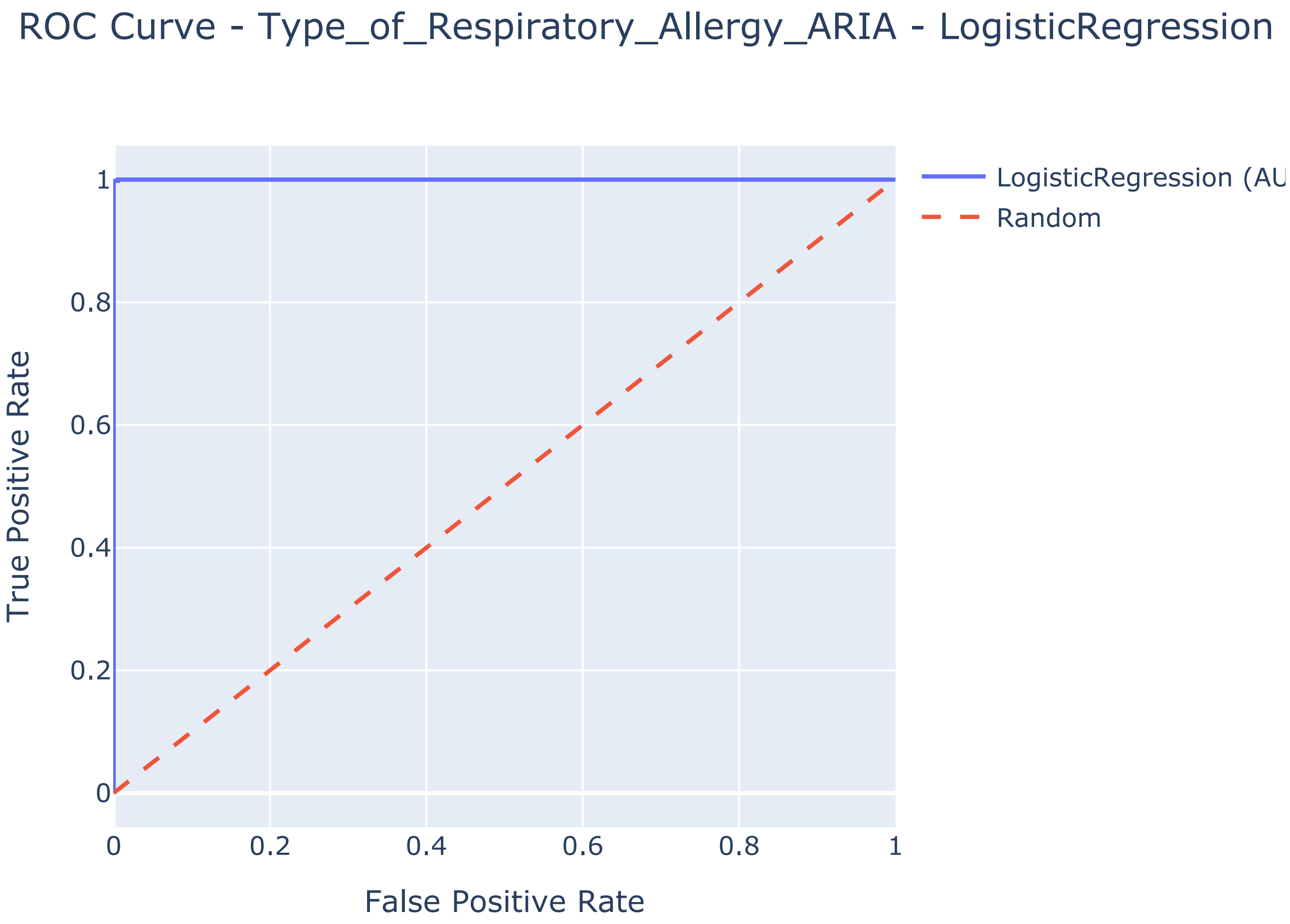
```
🔍 Target: Type_of_Respiratory_Allergy_ARIA | Model: RandomForest
📈 Accuracy: 0.9925
🎯 F1 (0): 0.9928 | F1 (1): 0.9921
📊 Precision: 0.9926 | AUC: 0.9983992551566081
📊 Confusion Matrix:
 [[557   0]
 [  0 506]]
```



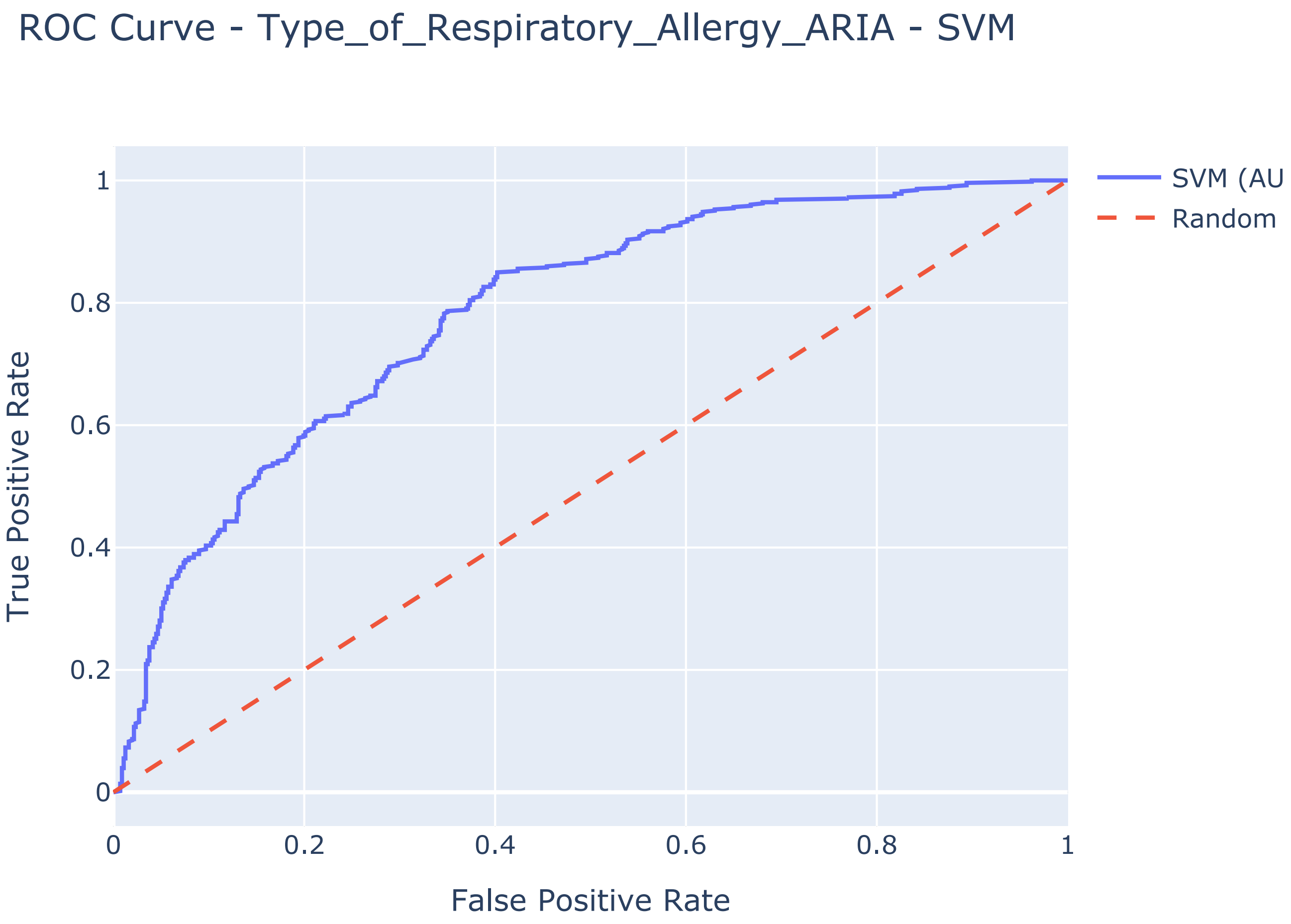
```
🔍 Target: Type_of_Respiratory_Allergy_ARIA | Model: XGBoost
📈 Accuracy: 0.9925
🎯 F1 (0): 0.9928 | F1 (1): 0.9921
📊 Precision: 0.9926 | AUC: 0.9990056022408964
📊 Confusion Matrix:
 [[557   0]
 [  0 506]]
```



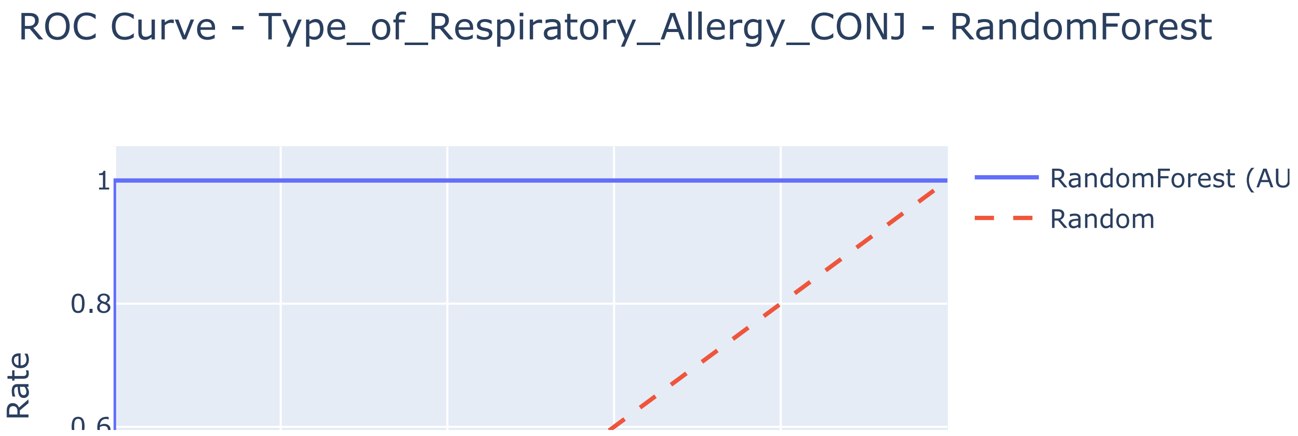
```
🔍 Target: Type_of_Respiratory_Allergy_ARIA | Model: LogisticRegression
📈 Accuracy: 0.9896
🎯 F1 (0): 0.9902 | F1 (1): 0.9891
📊 Precision: 0.9898 | AUC: 0.9996114081996434
📊 Confusion Matrix:
 [[557   0]
 [  4 502]]
```



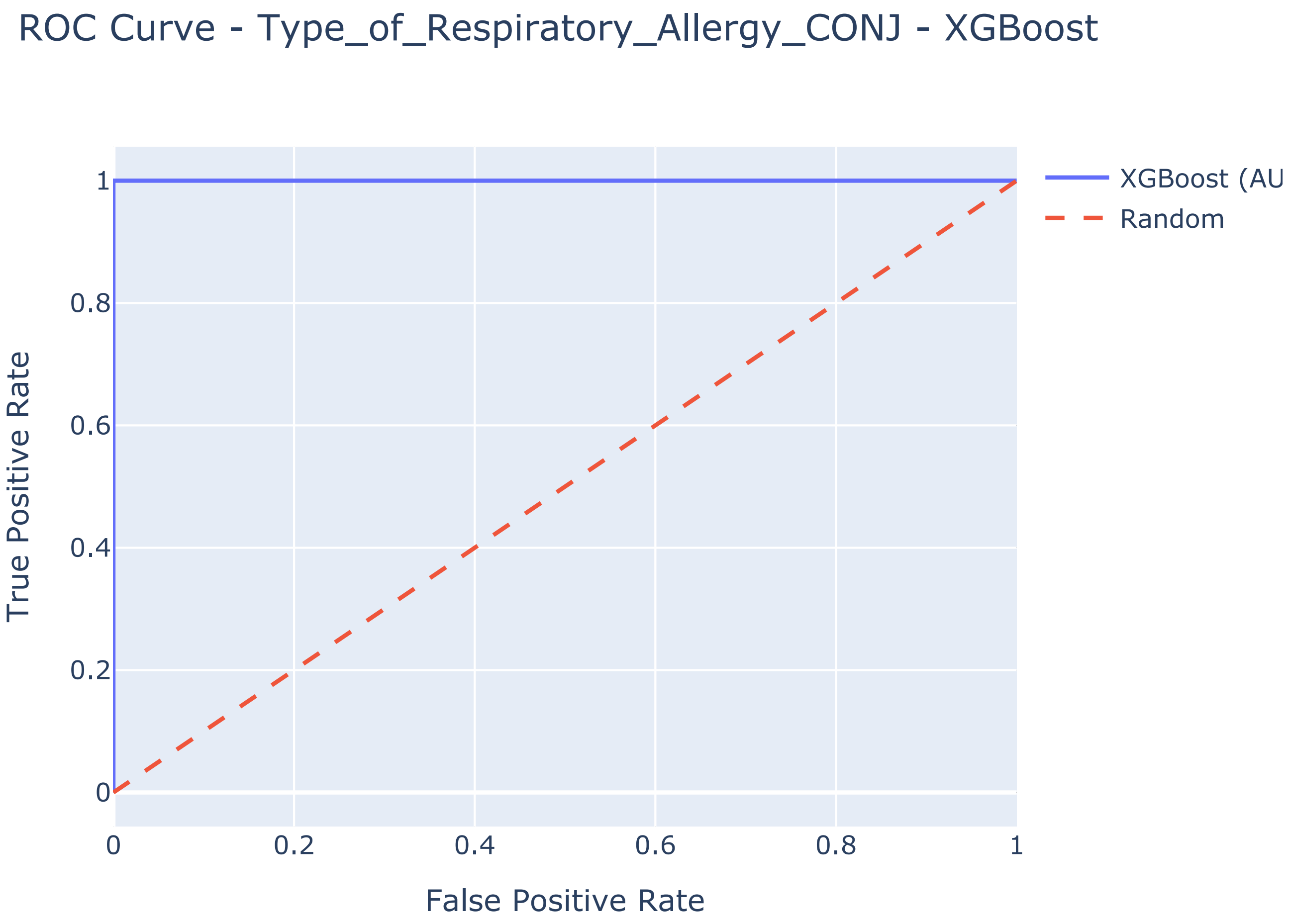
```
🔍 Target: Type_of_Respiratory_Allergy_ARIA | Model: SVM
📈 Accuracy: 0.6670
🎯 F1 (0): 0.6685 | F1 (1): 0.6649
📊 Precision: 0.6707 | AUC: 0.7262861599185129
📊 Confusion Matrix:
 [[380 177]
 [148 358]]
```



```
🔍 Target: Type_of_Respiratory_Allergy_CONJ | Model: RandomForest
📈 Accuracy: 0.9915
🎯 F1 (0): 0.9943 | F1 (1): 0.9836
📊 Precision: 0.9916 | AUC: 0.9996567544035898
📊 Confusion Matrix:
 [[787   0]
 [  0 276]]
```

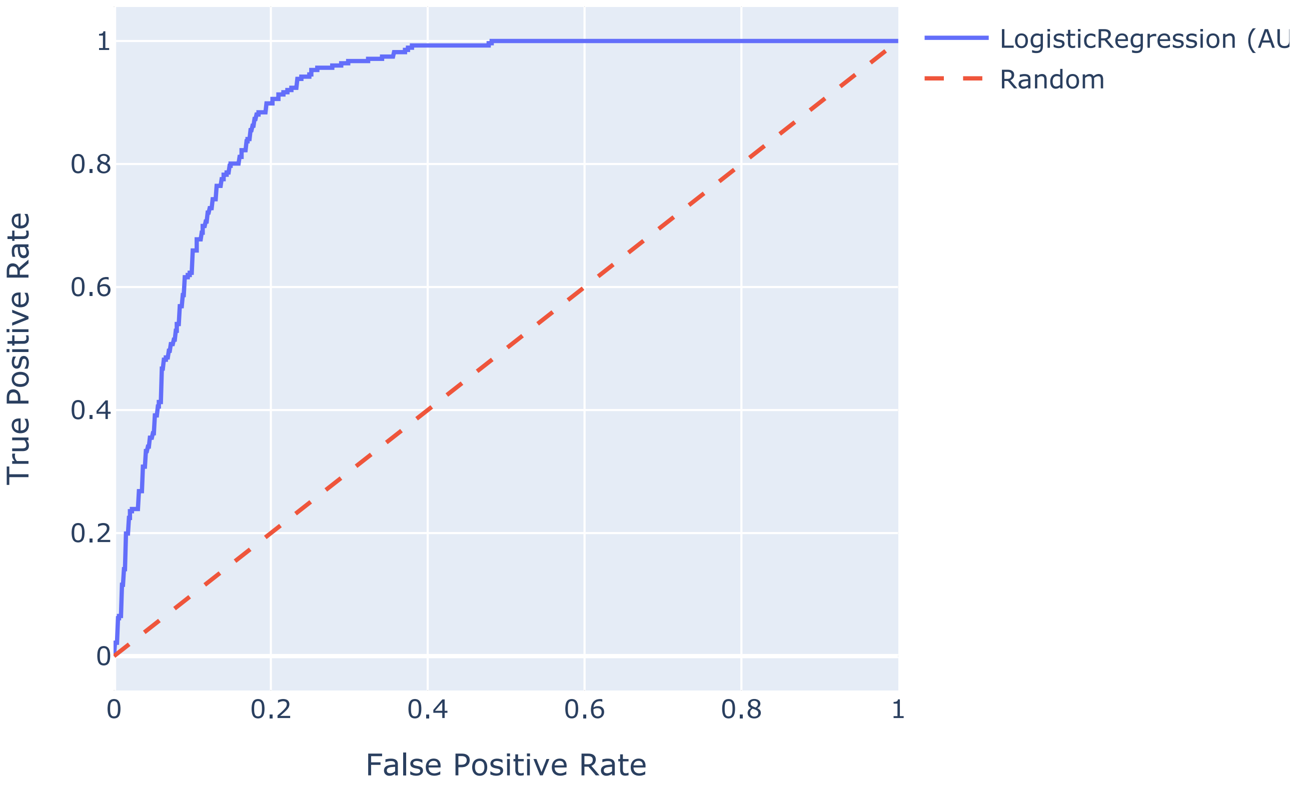


```
🔍 Target: Type_of_Respiratory_Allergy_CONJ | Model: XGBoost
📈 Accuracy: 0.9991
🎯 F1 (0): 0.9994 | F1 (1): 0.9982
📊 Precision: 0.9991 | AUC: 0.9996718237224567
📊 Confusion Matrix:
 [[787   0]
 [  0 276]]
```



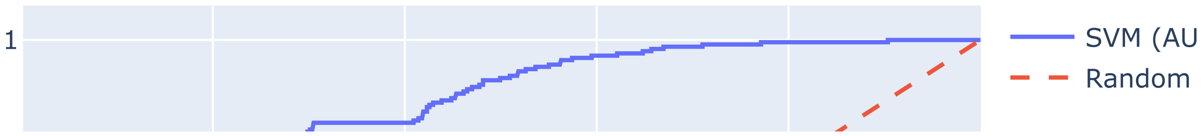
```
🔍 Target: Type_of_Respiratory_Allergy_CONJ | Model: LogisticRegression
📈 Accuracy: 0.7912
🎯 F1 (0): 0.8541 | F1 (1): 0.6295
📊 Precision: 0.8060 | AUC: 0.848039571773749
📊 Confusion Matrix:
 [[700  87]
 [ 88 188]]
```

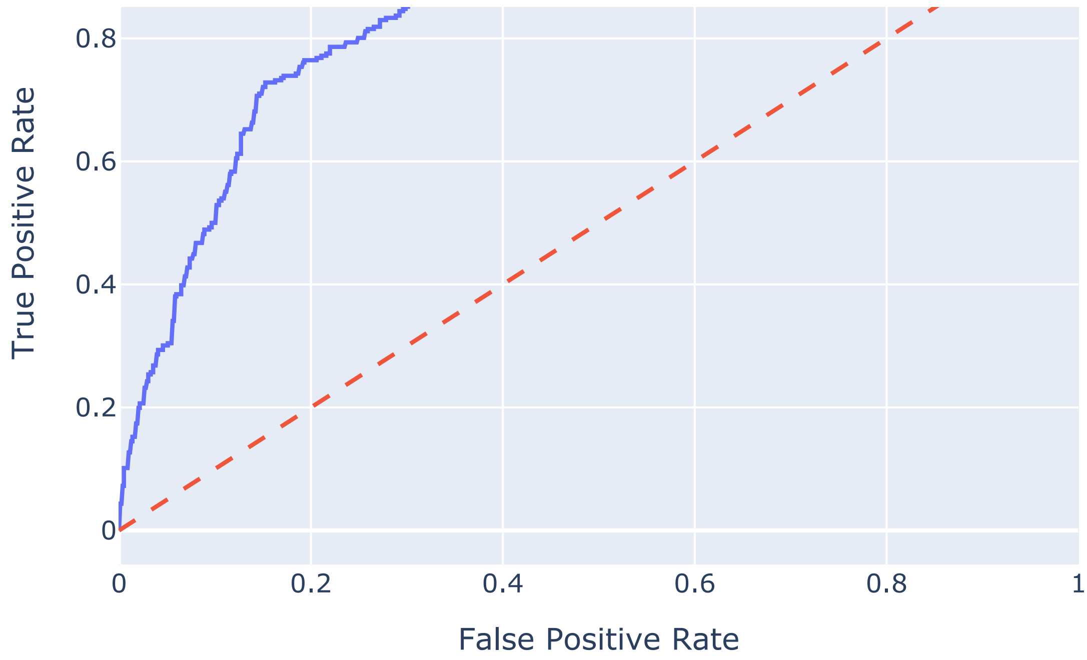
ROC Curve - Type_of_Respiratory_Allergy_CONJ - LogisticRegression



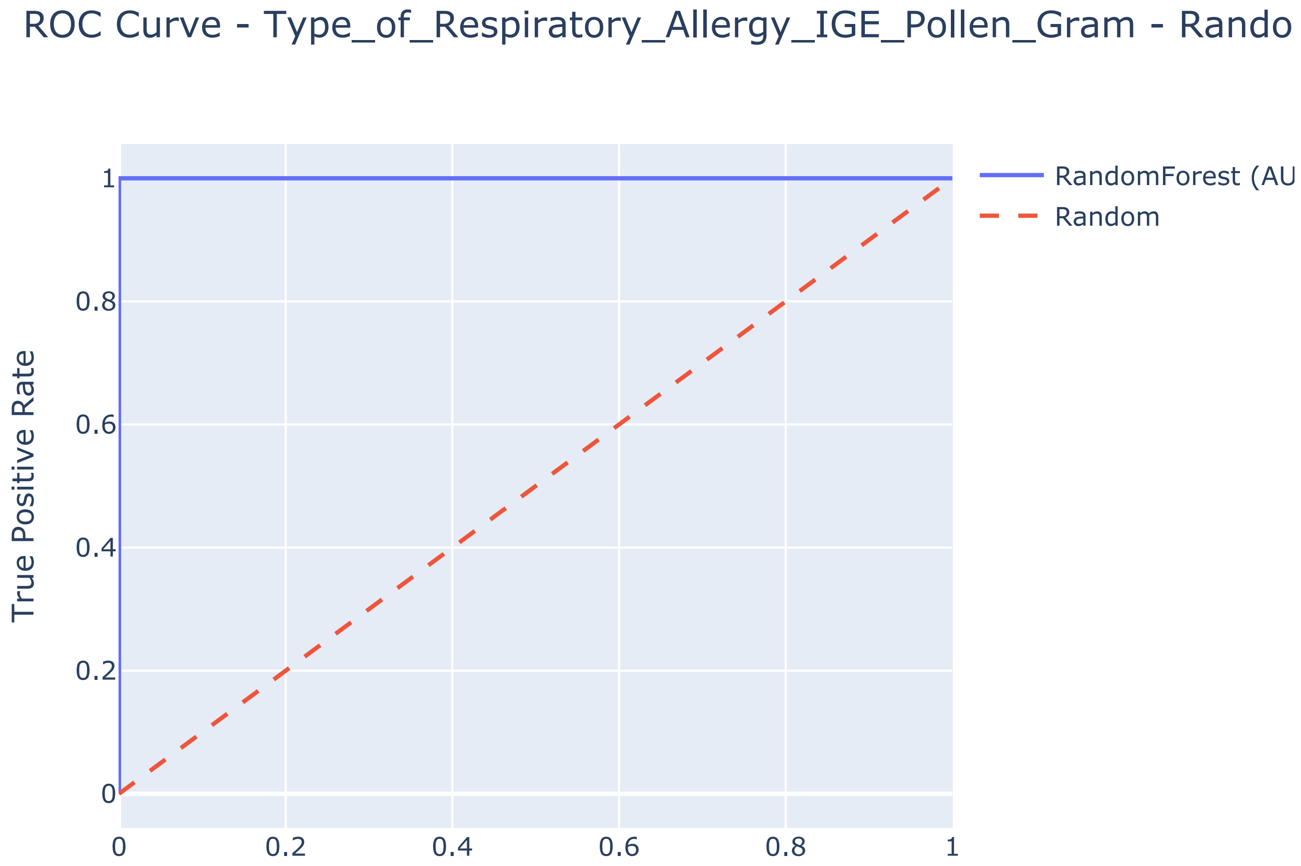
```
🔍 Target: Type_of_Respiratory_Allergy_CONJ | Model: SVM
📈 Accuracy: 0.6877
🎯 F1 (0): 0.7453 | F1 (1): 0.5949
📊 Precision: 0.8104 | AUC: 0.7328171125639481
📊 Confusion Matrix:
 [[785   2]
 [262  14]]
```

ROC Curve - Type_of_Respiratory_Allergy_CONJ - SVM





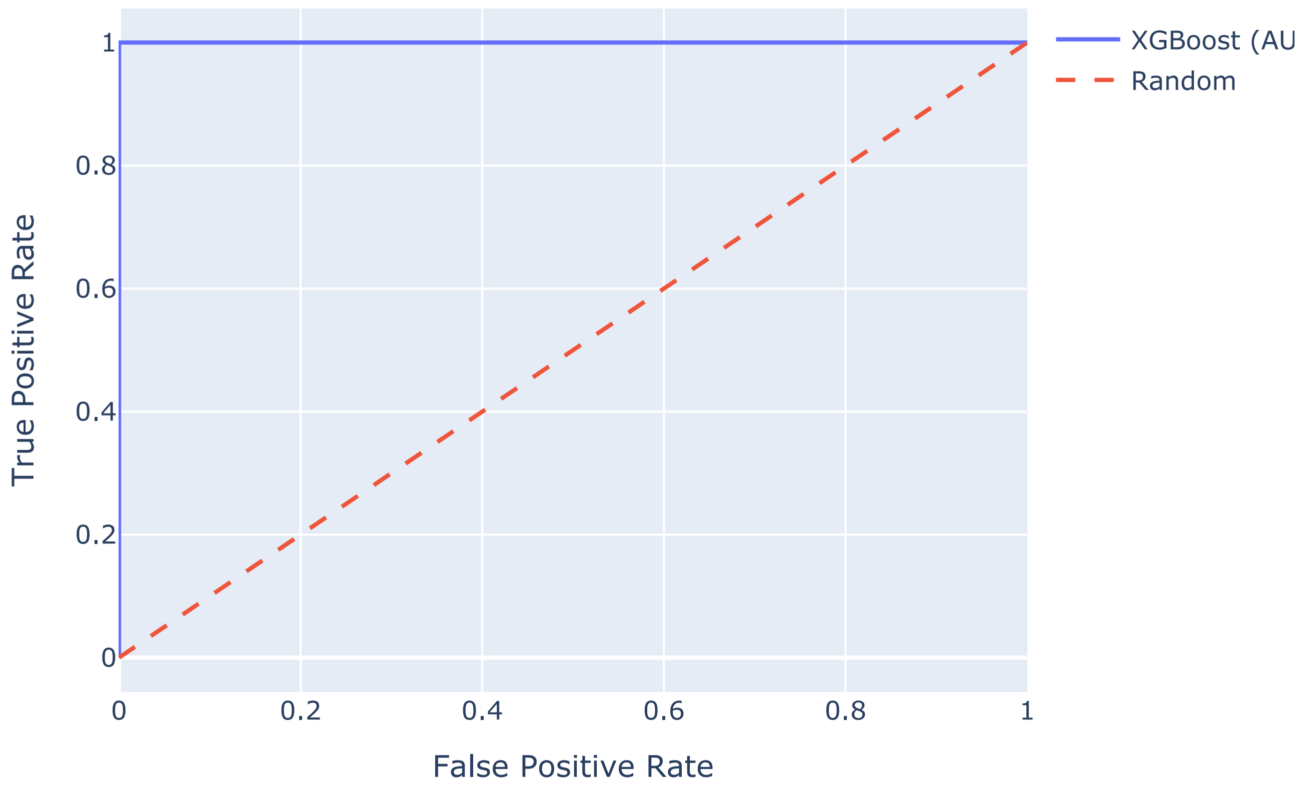
```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Pollen_Gram | Model: RandomFores
📈 Accuracy: 0.8279
🎯 F1 (0): 0.7549 | F1 (1): 0.8672
📊 Precision: 0.8277 | AUC: 0.9112214398407023
📊 Confusion Matrix:
 [[379   0]
 [  0 684]]
```



False Positive Rate

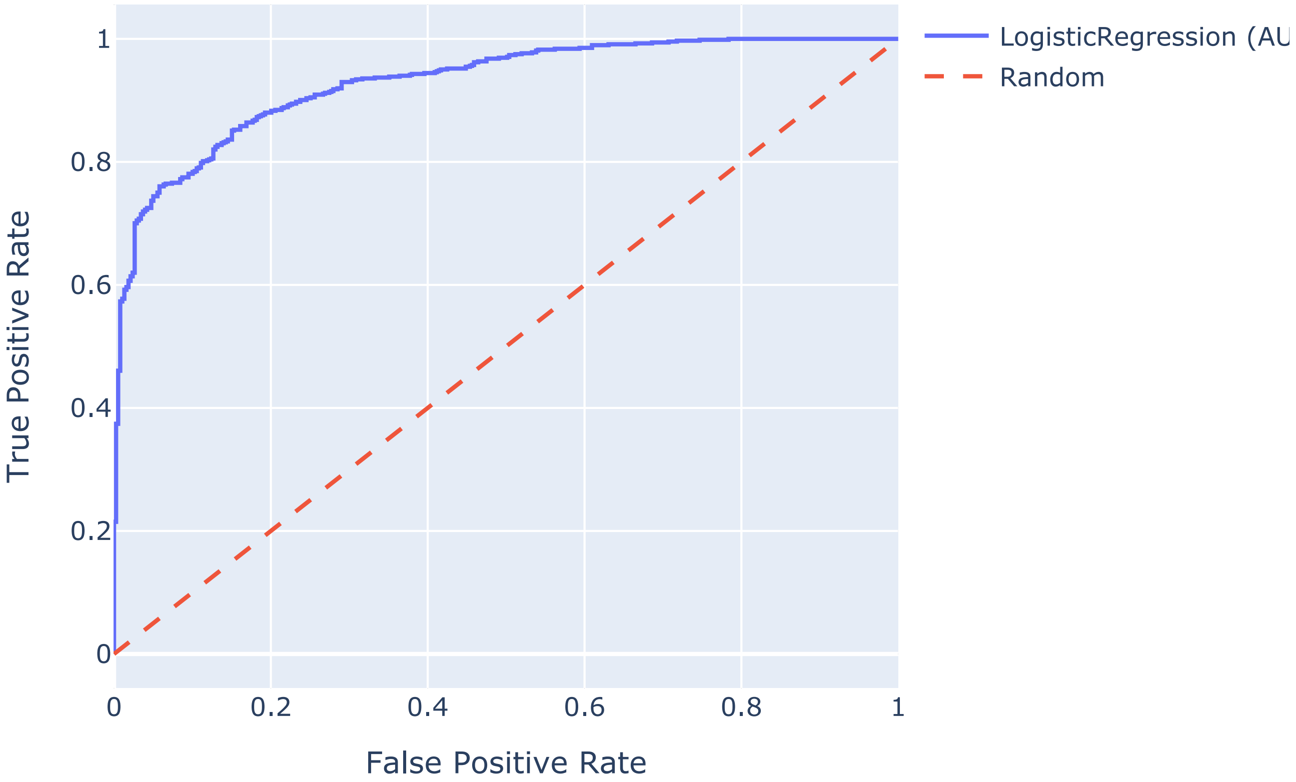
```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Pollen_Gram | Model: XGBoost
📈 Accuracy: 0.8513
🎯 F1 (0): 0.7940 | F1 (1): 0.8835
📊 Precision: 0.8530 | AUC: 0.9335121280251851
📊 Confusion Matrix:
 [[379   0]
 [  0 684]]
```

ROC Curve - Type_of_Respiratory_Allergy_IGE_Pollen_Gram - XGBoc



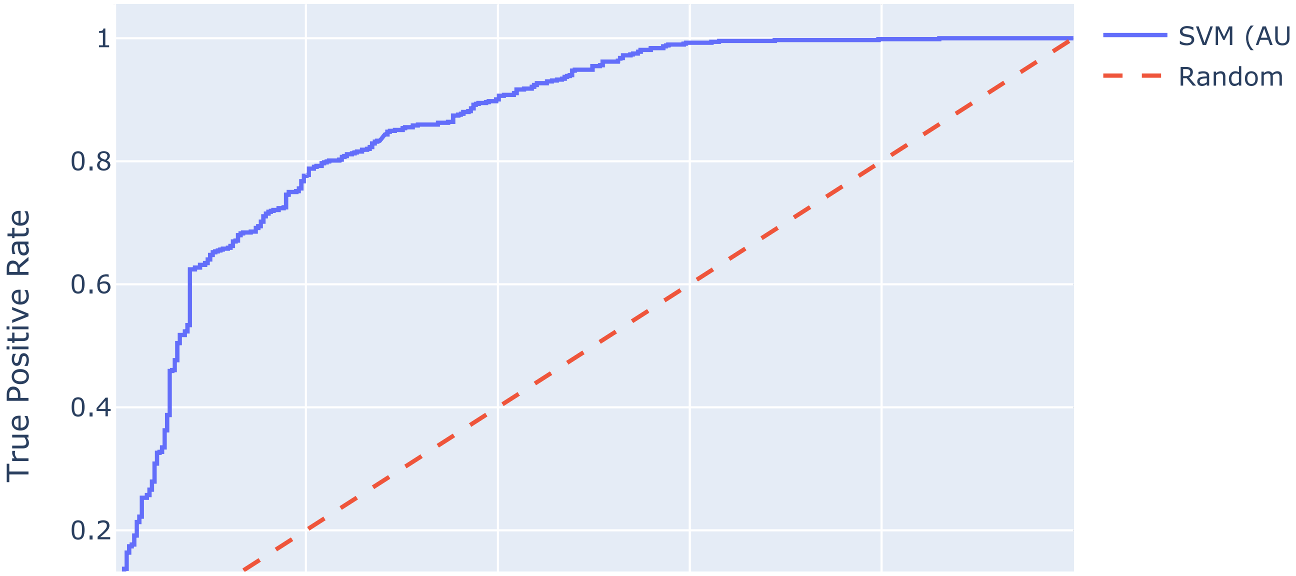
```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Pollen_Gram | Model: LogisticReg
📈 Accuracy: 0.7921
🎯 F1 (0): 0.7342 | F1 (1): 0.8292
📊 Precision: 0.8063 | AUC: 0.8787284642968449
📊 Confusion Matrix:
 [[315  64]
 [ 94 590]]
```

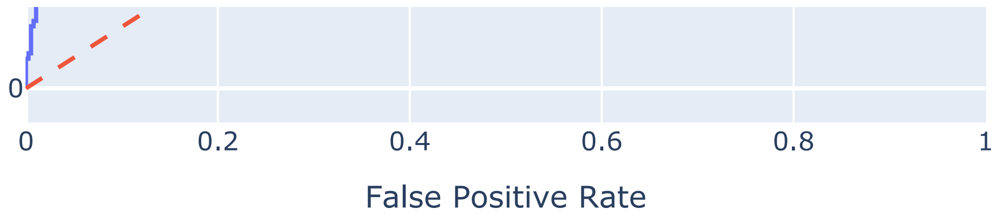
ROC Curve - Type_of_Respiratory_Allergy_IGE_Pollen_Gram - Logisti



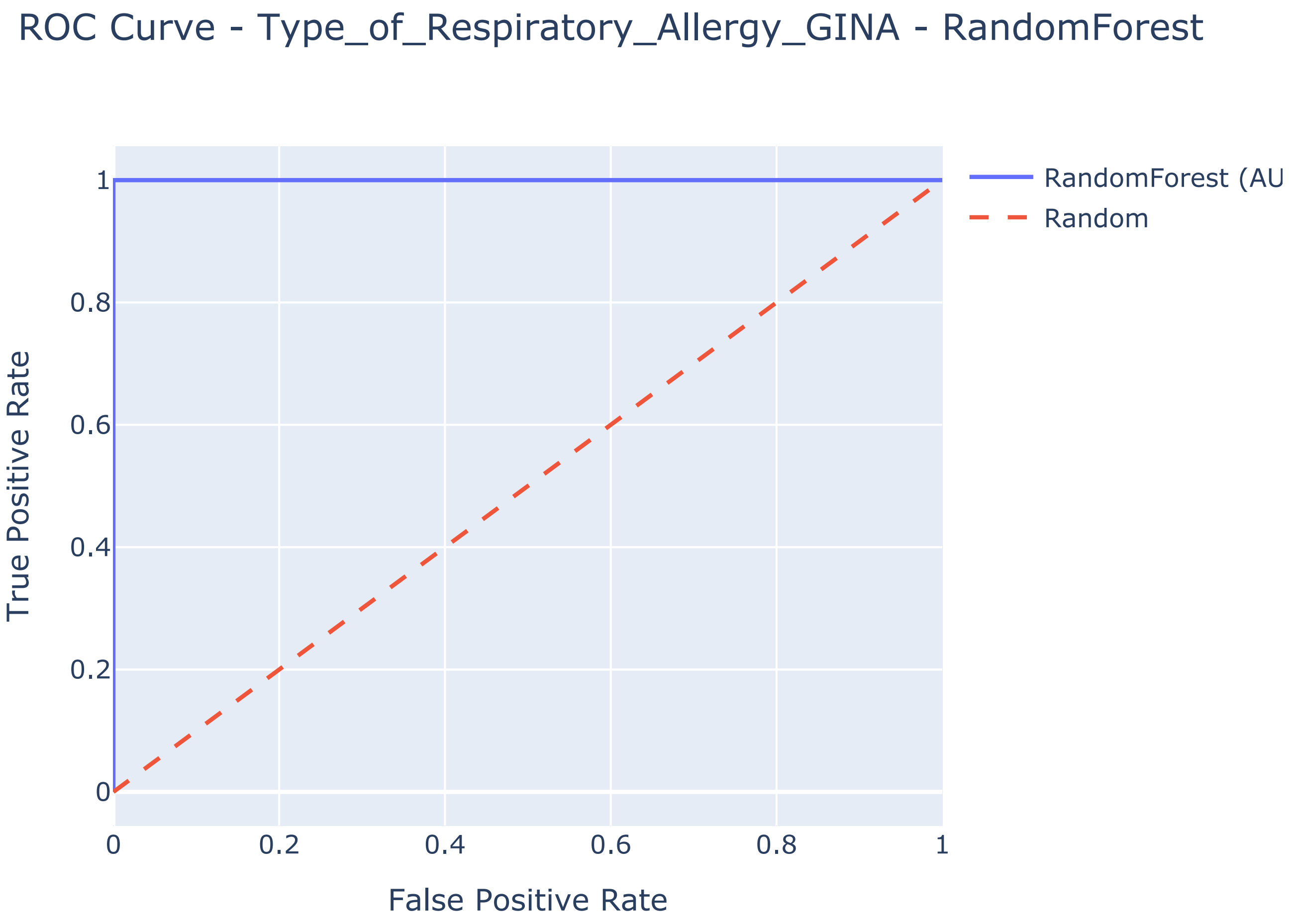
```
🔍 Target: Type_of_Respiratory_Allergy_IGE_Pollen_Gram | Model: SVM
📈 Accuracy: 0.7602
🎯 F1 (0): 0.7050 | F1 (1): 0.7974
📊 Precision: 0.7852 | AUC: 0.8563112782994328
📊 Confusion Matrix:
 [[291  88]
 [136 548]]
```

ROC Curve - Type_of_Respiratory_Allergy_IGE_Pollen_Gram - SVM

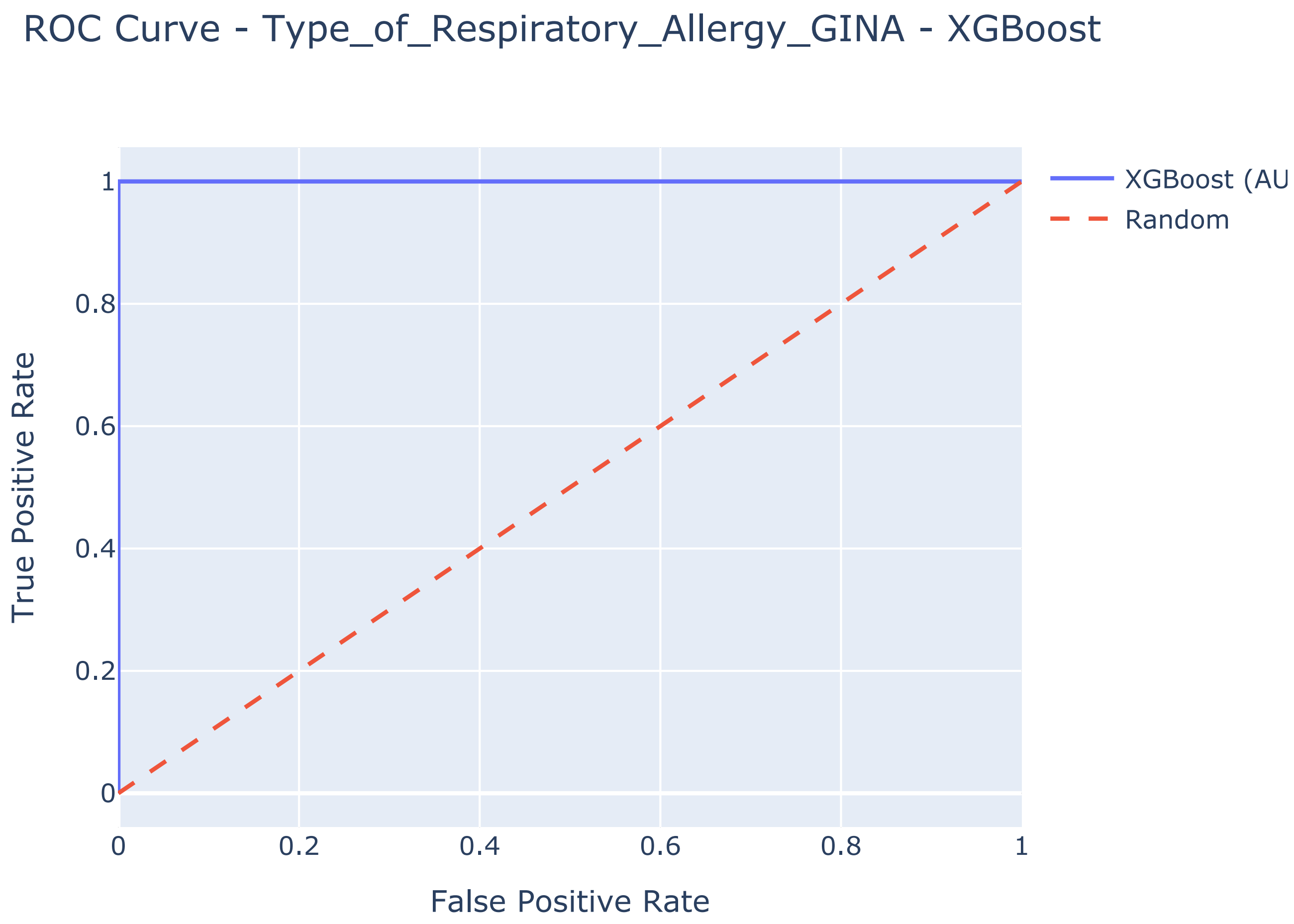




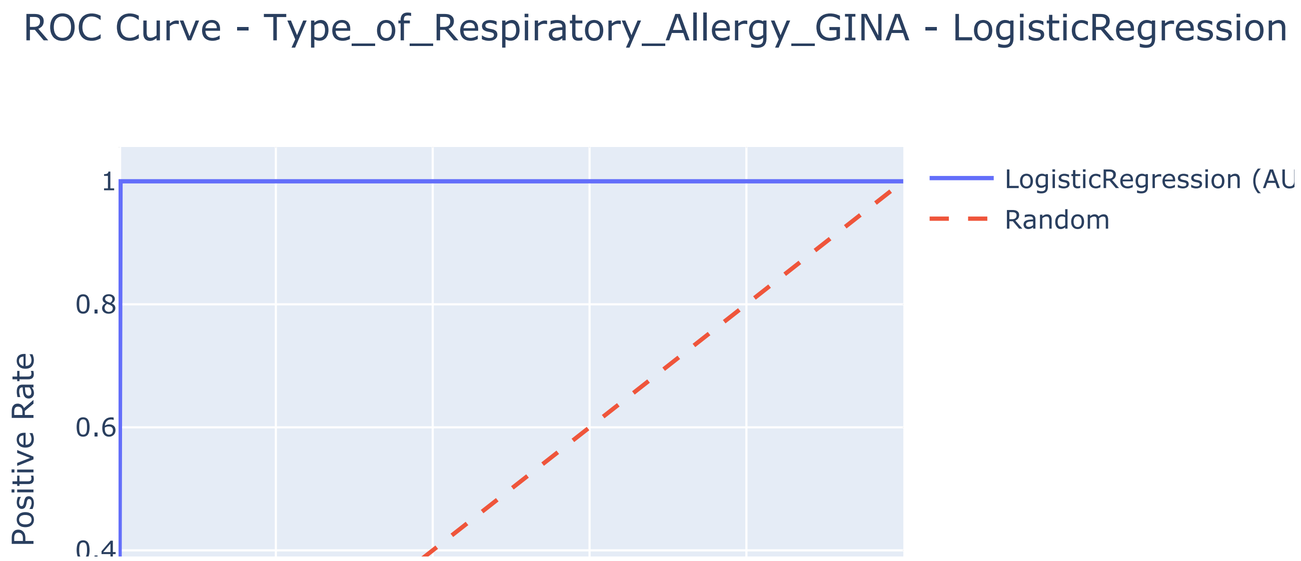
```
🔍 Target: Type_of_Respiratory_Allergy_GINA | Model: RandomForest
📈 Accuracy: 0.9906
🎯 F1 (0): 0.9905 | F1 (1): 0.9907
📊 Precision: 0.9907 | AUC: 0.9991129241129242
📊 Confusion Matrix:
 [[520   0]
 [  0 543]]
```



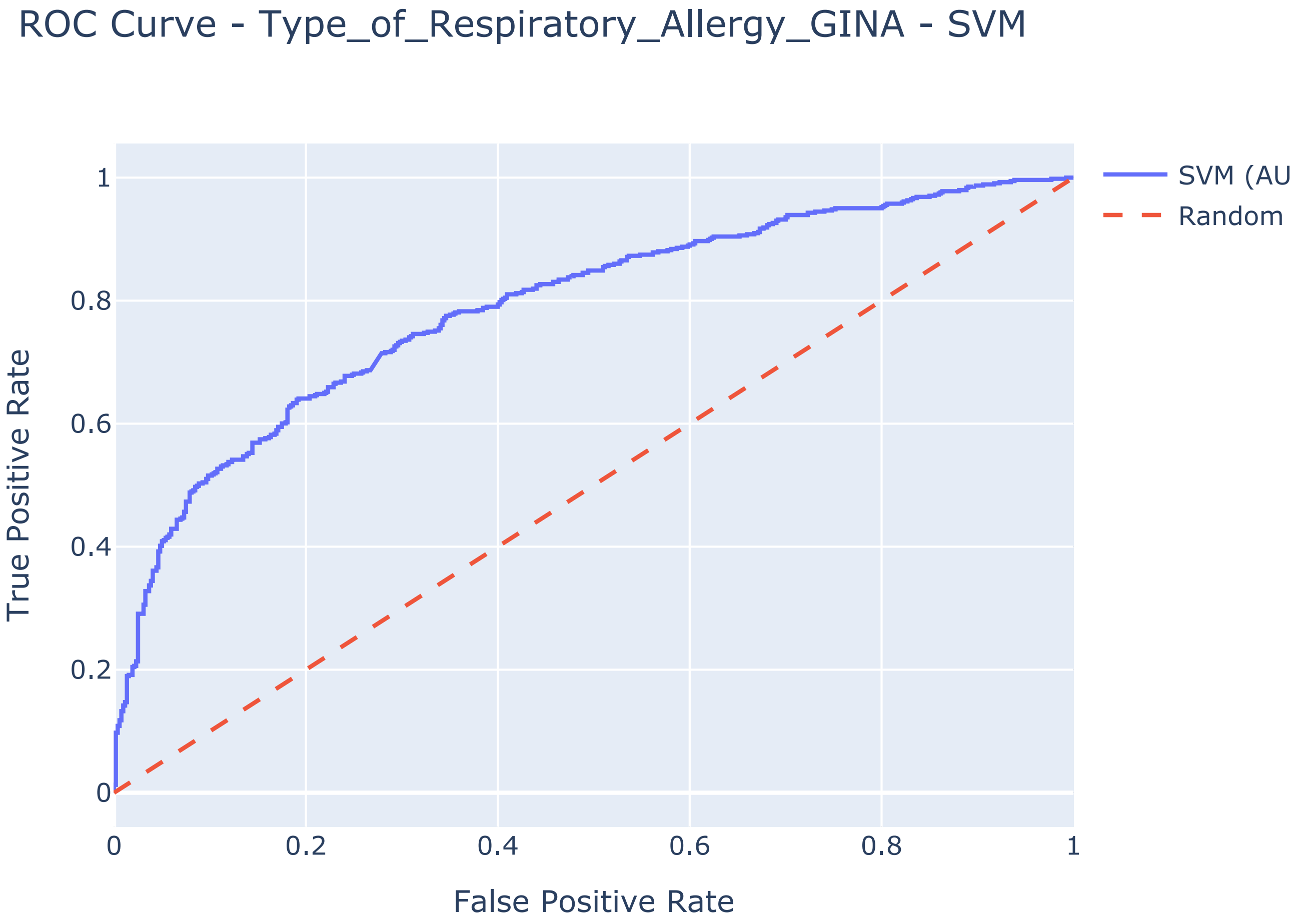
```
🔍 Target: Type_of_Respiratory_Allergy_GINA | Model: XGBoost
📈 Accuracy: 0.9887
🎯 F1 (0): 0.9887 | F1 (1): 0.9887
📊 Precision: 0.9890 | AUC: 0.9993945868945868
📊 Confusion Matrix:
 [[520   0]
 [  0 543]]
```



```
🔍 Target: Type_of_Respiratory_Allergy_GINA | Model: LogisticRegression
📈 Accuracy: 0.9877
🎯 F1 (0): 0.9877 | F1 (1): 0.9878
📊 Precision: 0.9881 | AUC: 0.9996082621082621
📊 Confusion Matrix:
 [[519   1]
 [  4 539]]
```



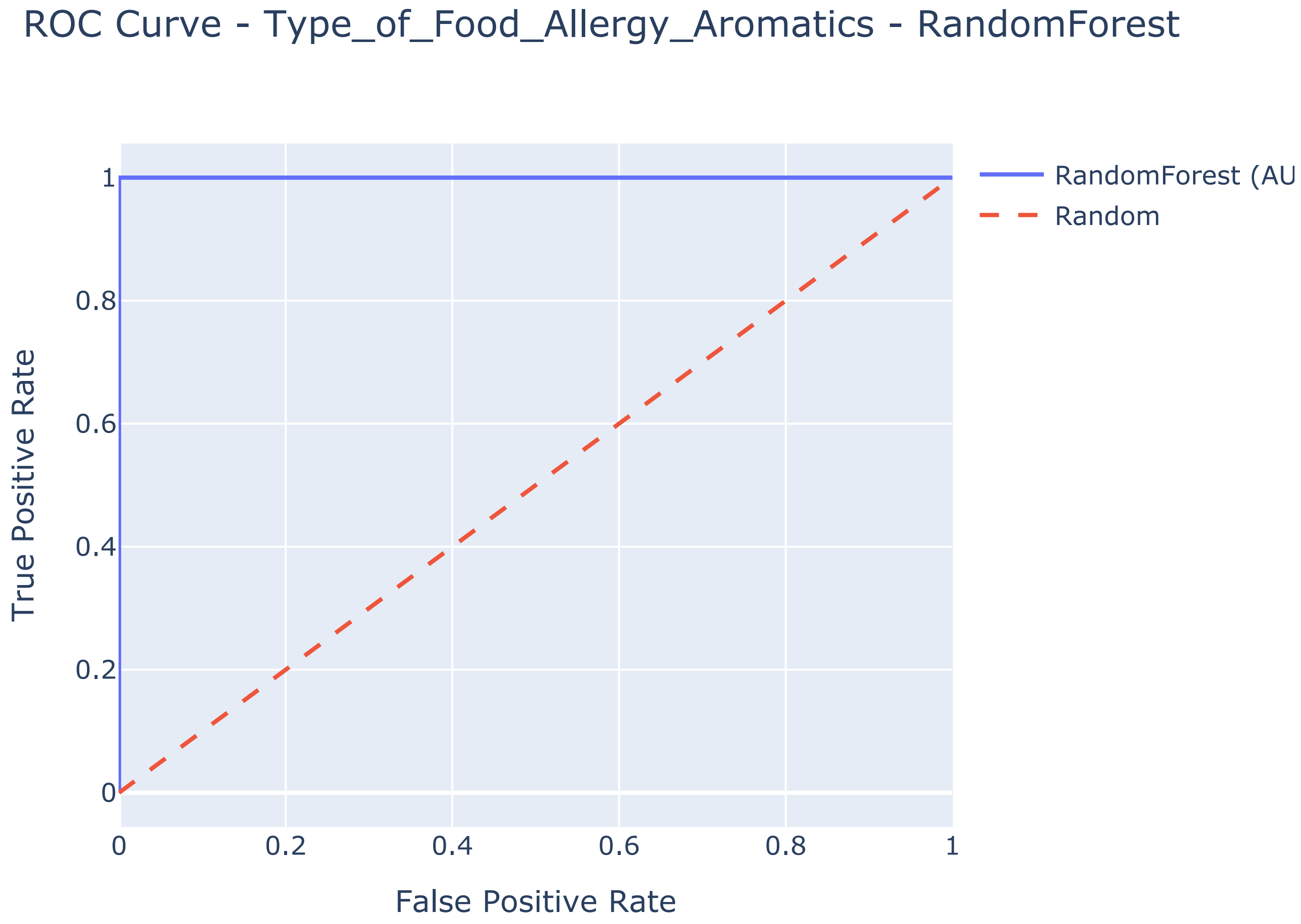
```
🔍 Target: Type_of_Respiratory_Allergy_GINA | Model: SVM
📈 Accuracy: 0.6697
🎯 F1 (0): 0.6762 | F1 (1): 0.6614
📊 Precision: 0.6733 | AUC: 0.7325848225848225
📊 Confusion Matrix:
 [[387 133]
 [173 370]]
```



```
import pandas as pd
import numpy as np
```

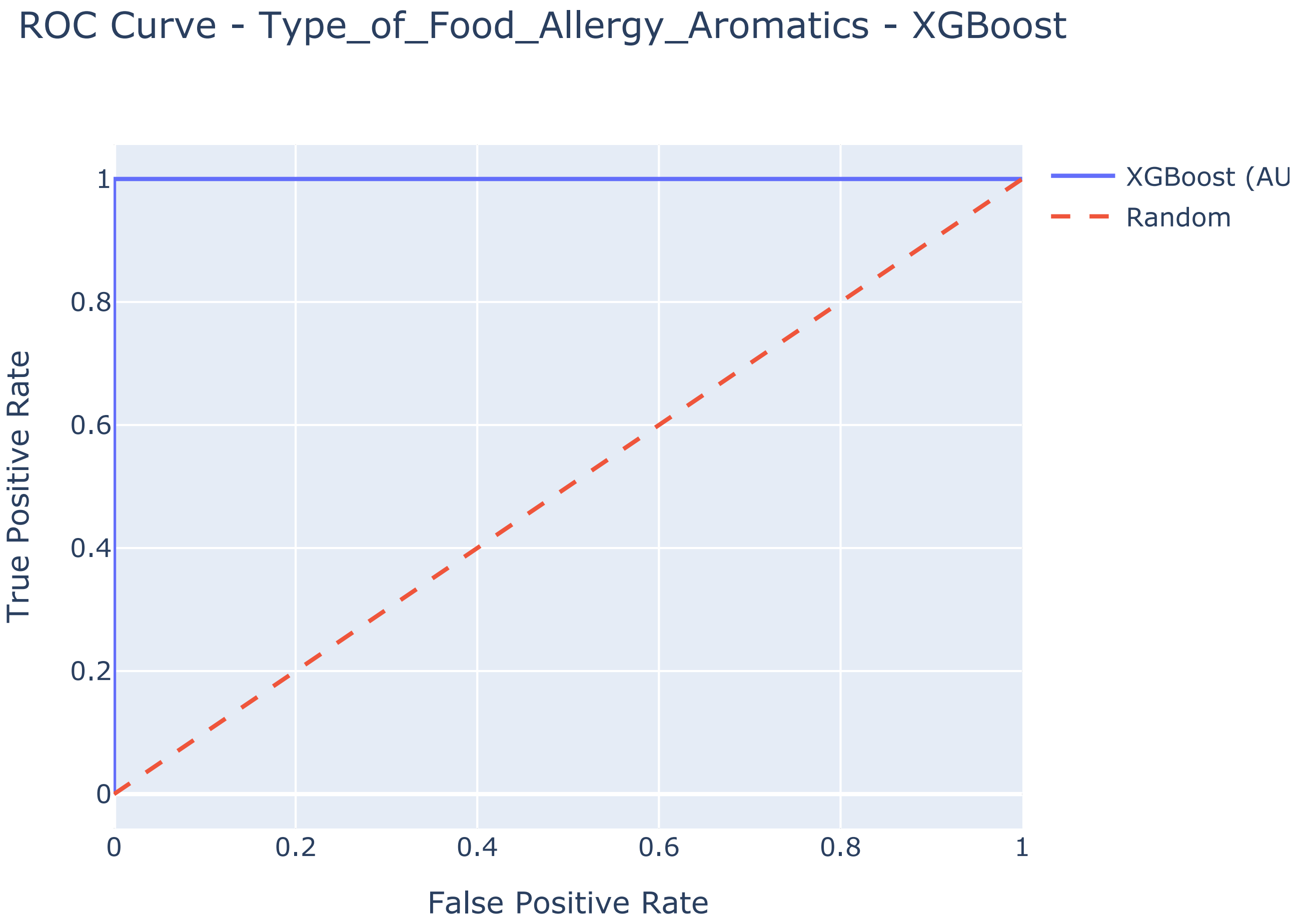
```
from sklearn.model_selection import StratifiedKFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.metrics import (
    f1_score, accuracy_score, recall_score,
    precision_score,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

V1_food = V1[V1["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=V1_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 = V1_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_V1_food.csv", index=False)
```

```
🔍 Target: Type_of_Food_Allergy_Aromatics | Model: RandomForest
📈 Accuracy: 0.9776
🎯 F1 (0): 0.9887 | F1 (1): 0.0667
📊 Precision: 0.9662 | AUC: 0.8009273772204806
📊 Confusion Matrix:
 [[878   0]
 [  0  16]]
```

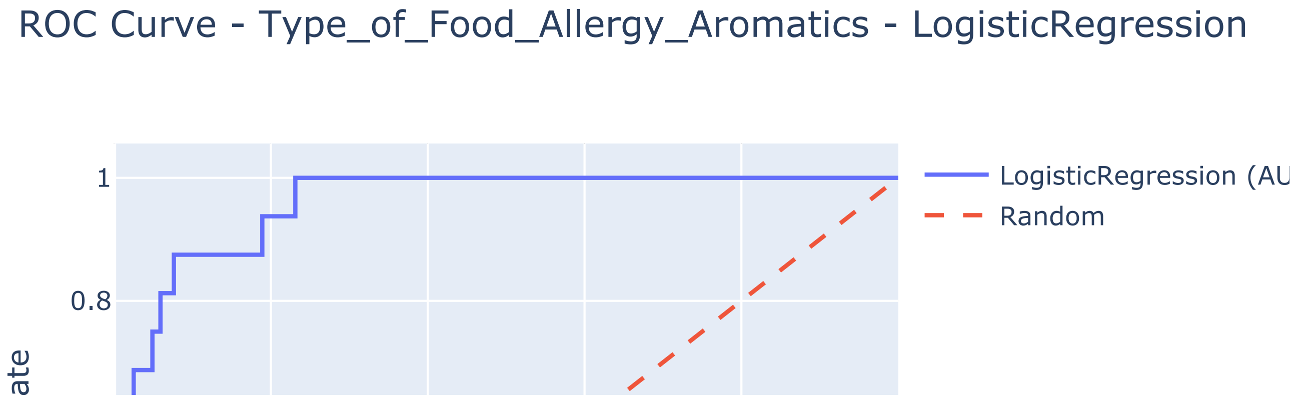


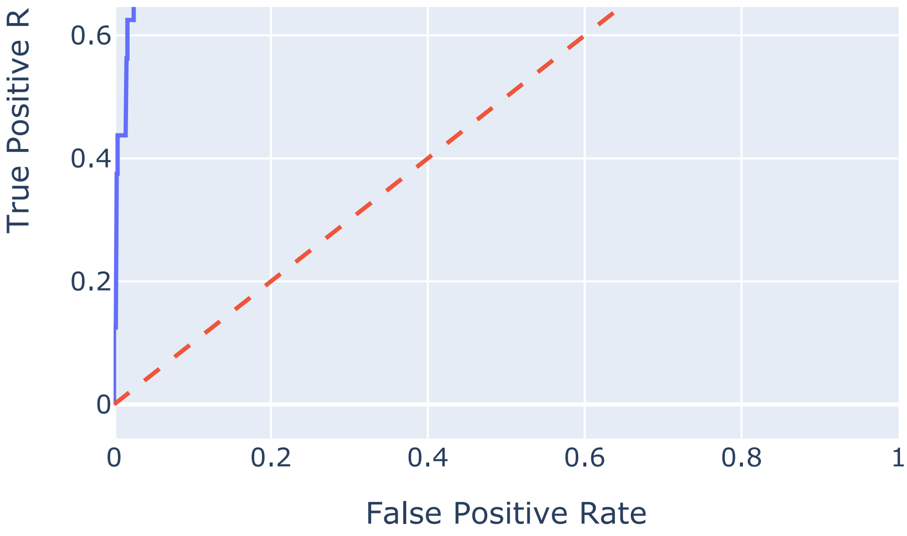
```
🔍 Target: Type_of_Food_Allergy_Aromatics | Model: XGBoost
📈 Accuracy: 0.9754
🎯 F1 (0): 0.9875 | F1 (1): 0.1900
```

```
📊 Precision: 0.9727 | AUC: 0.7526907001044931
📊 Confusion Matrix:
 [[878   0]
 [  0  16]]
```

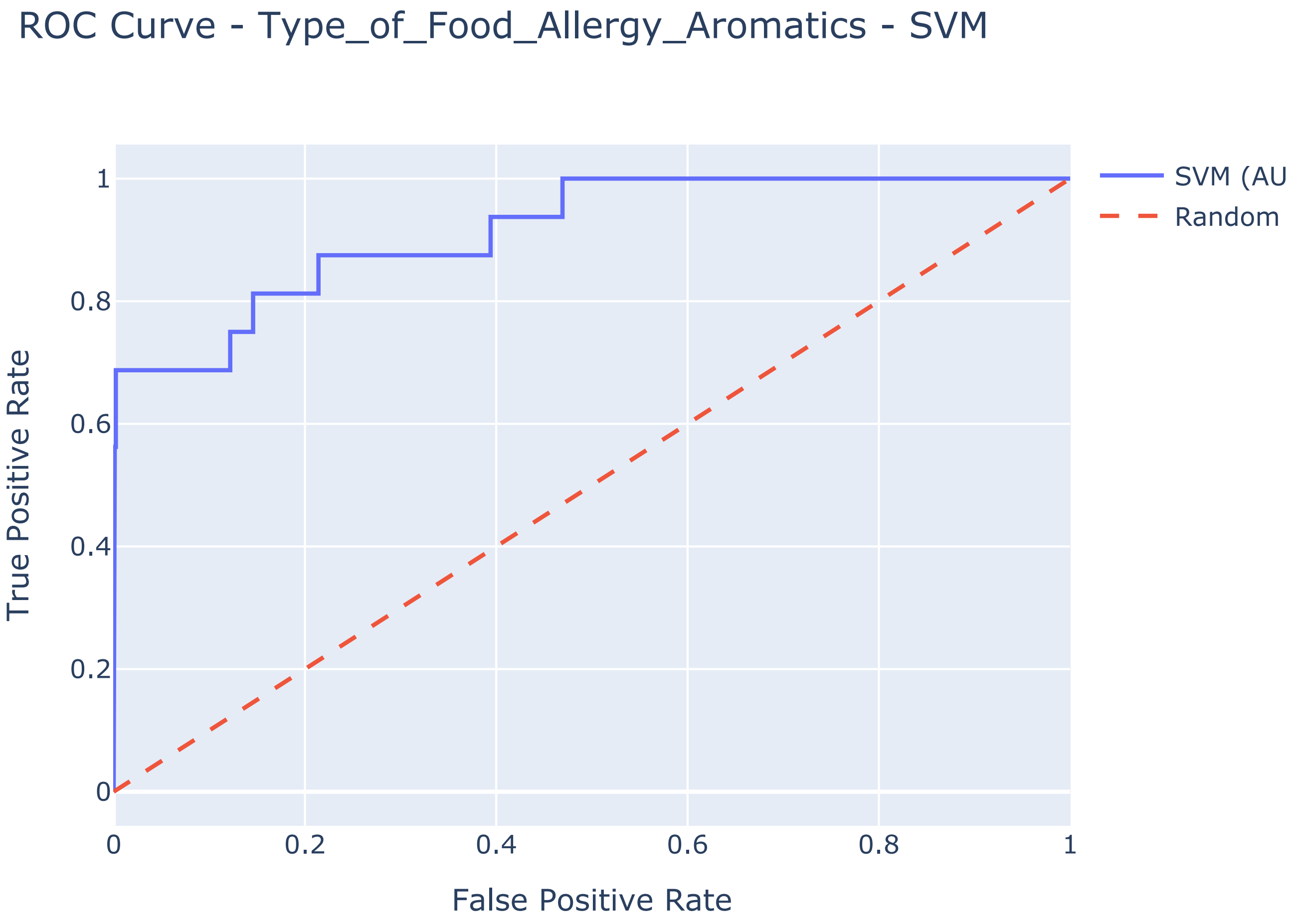


```
🔍 Target: Type_of_Food_Allergy_Aromatics | Model: LogisticRegression
📈 Accuracy: 0.9351
🎯 F1 (0): 0.9662 | F1 (1): 0.0832
📊 Precision: 0.9684 | AUC: 0.7120885579937305
📊 Confusion Matrix:
 [[878   0]
 [ 15   1]]
```

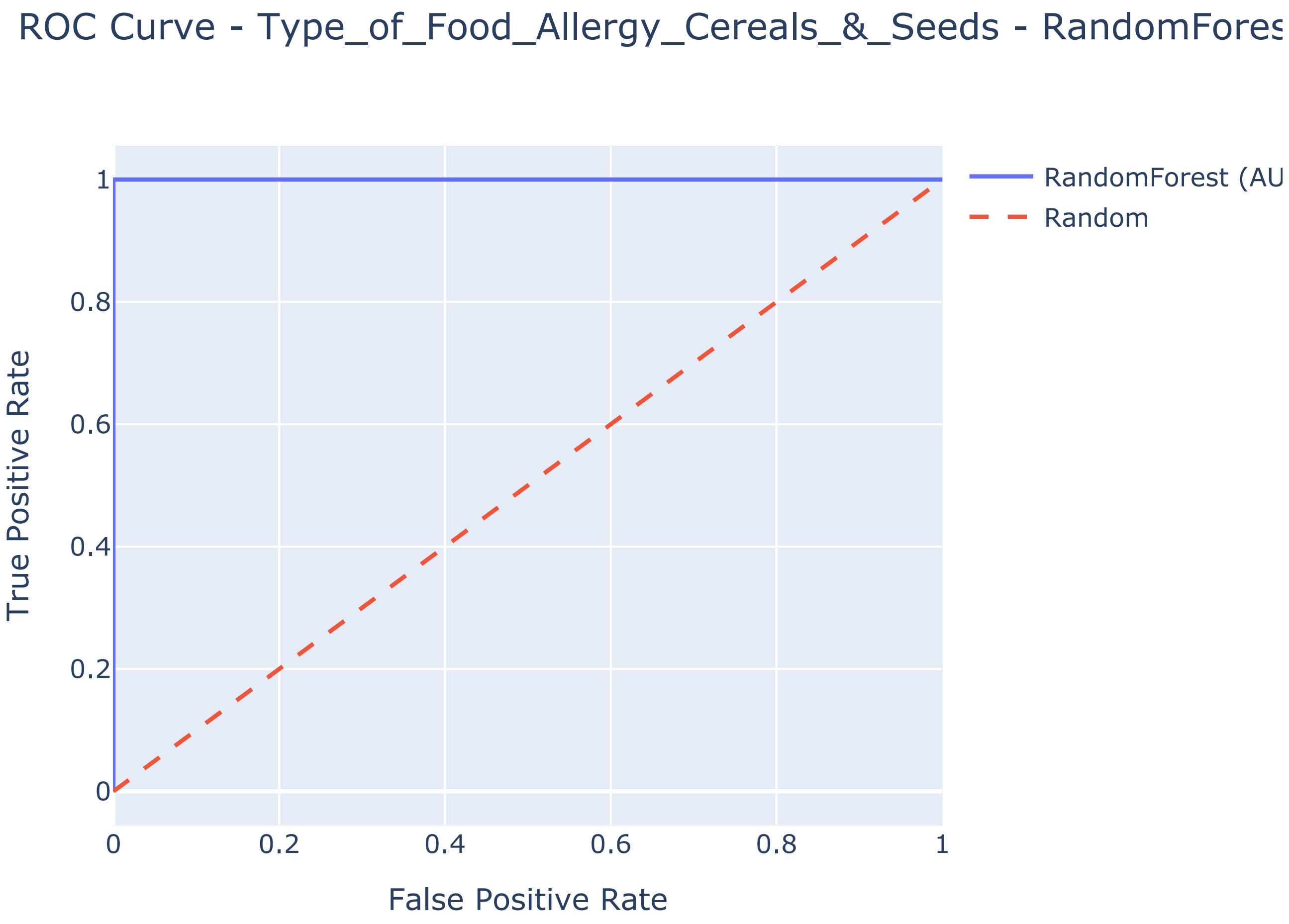




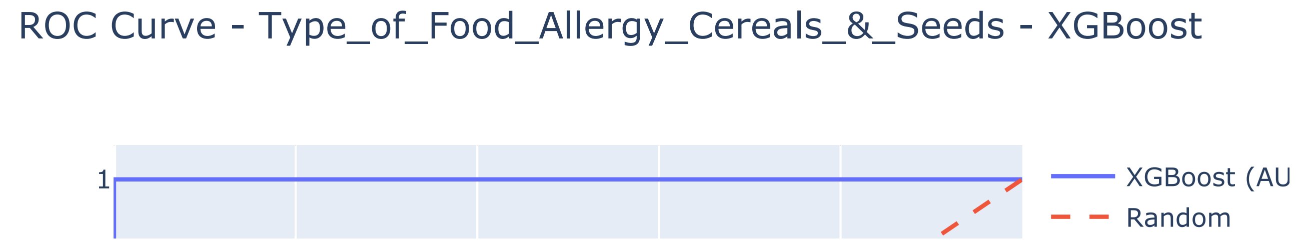
```
🔍 Target: Type_of_Food_Allergy_Aromatics | Model: SVM
📈 Accuracy: 0.7998
🎯 F1 (0): 0.8872 | F1 (1): 0.0868
📊 Precision: 0.9737 | AUC: 0.7221068443051202
📊 Confusion Matrix:
 [[878   0]
 [ 16   0]]
```

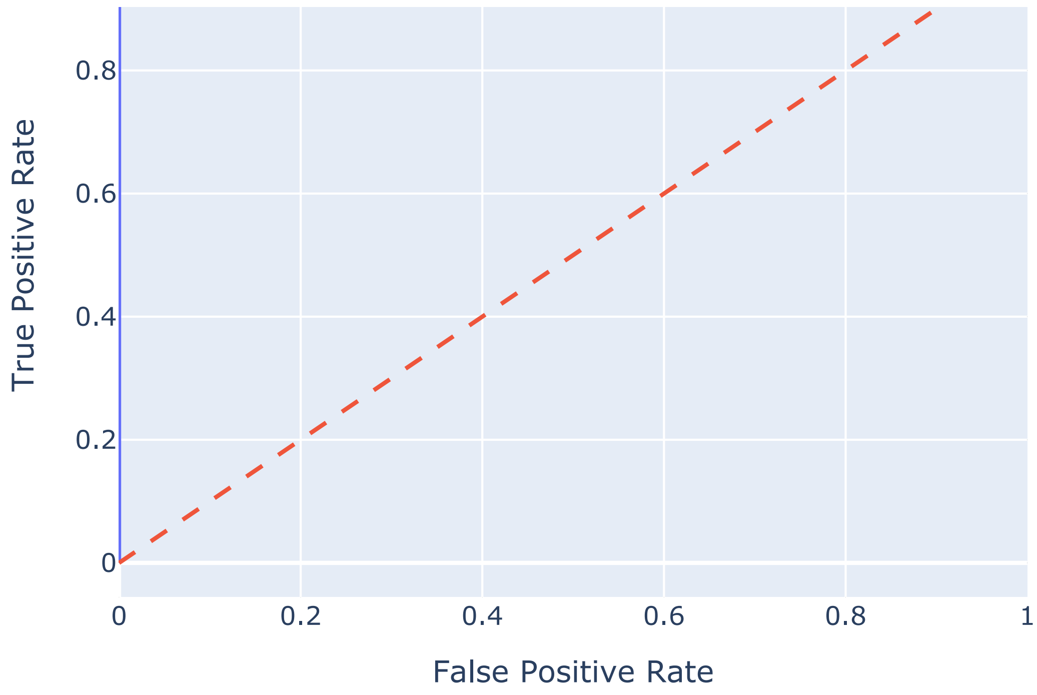


```
🔍 Target: Type_of_Food_Allergy_Cereals_&_Seeds | Model: RandomForest
📈 Accuracy: 0.9608
🎯 F1 (0): 0.9800 | F1 (1): 0.0000
📊 Precision: 0.9403 | AUC: 0.7197986278178738
📊 Confusion Matrix:
 [[867   0]
 [  0  27]]
```

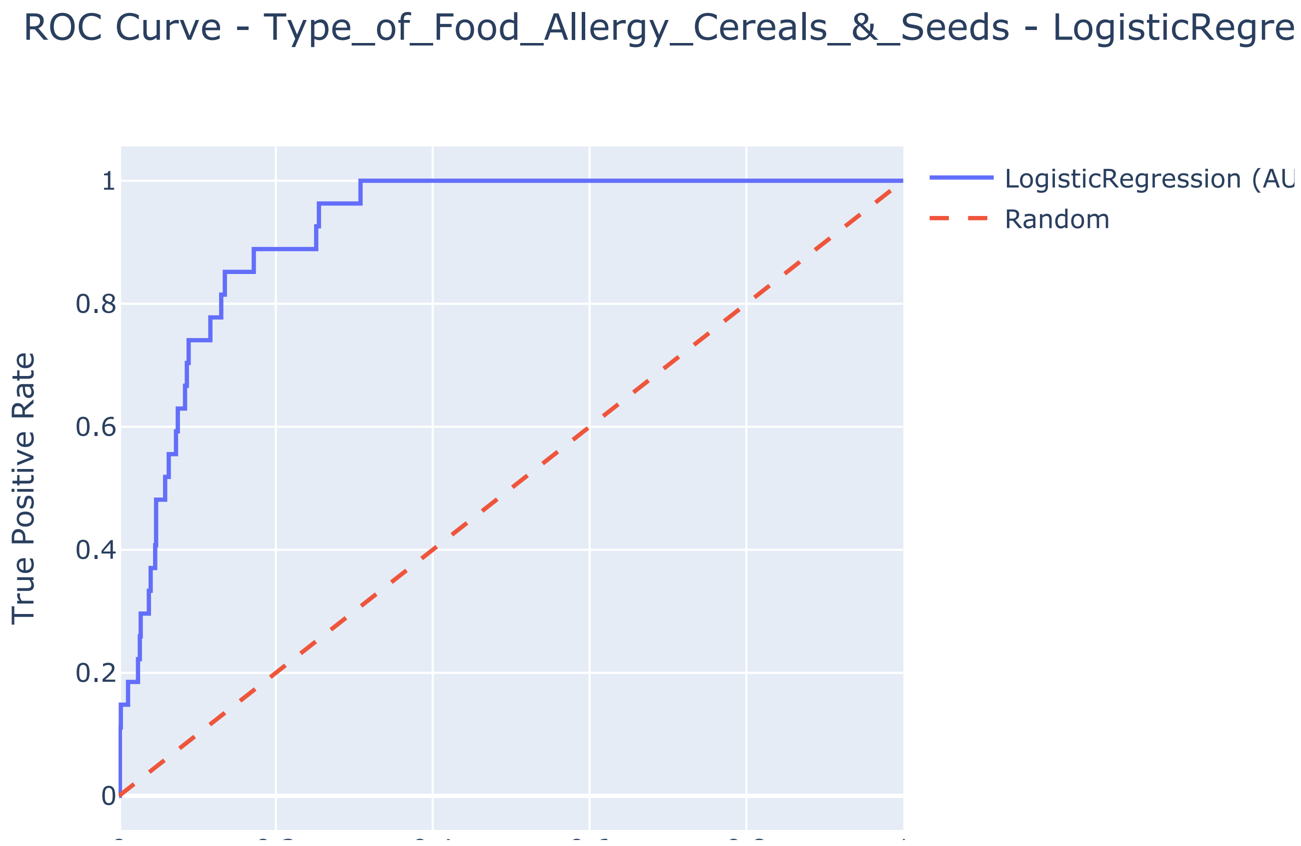

ROC Curve - Type_of_Food_Allergy_Cereals_&_Seeds - RandomFores

```
🔍 Target: Type_of_Food_Allergy_Cereals_&_Seeds | Model: XGBoost
📈 Accuracy: 0.9507
🎯 F1 (0): 0.9747 | F1 (1): 0.0000
📊 Precision: 0.9400 | AUC: 0.6871380201372181
📊 Confusion Matrix:
 [[867   0]
 [  0  27]]
```


ROC Curve - Type_of_Food_Allergy_Cereals_&_Seeds - XGBoost

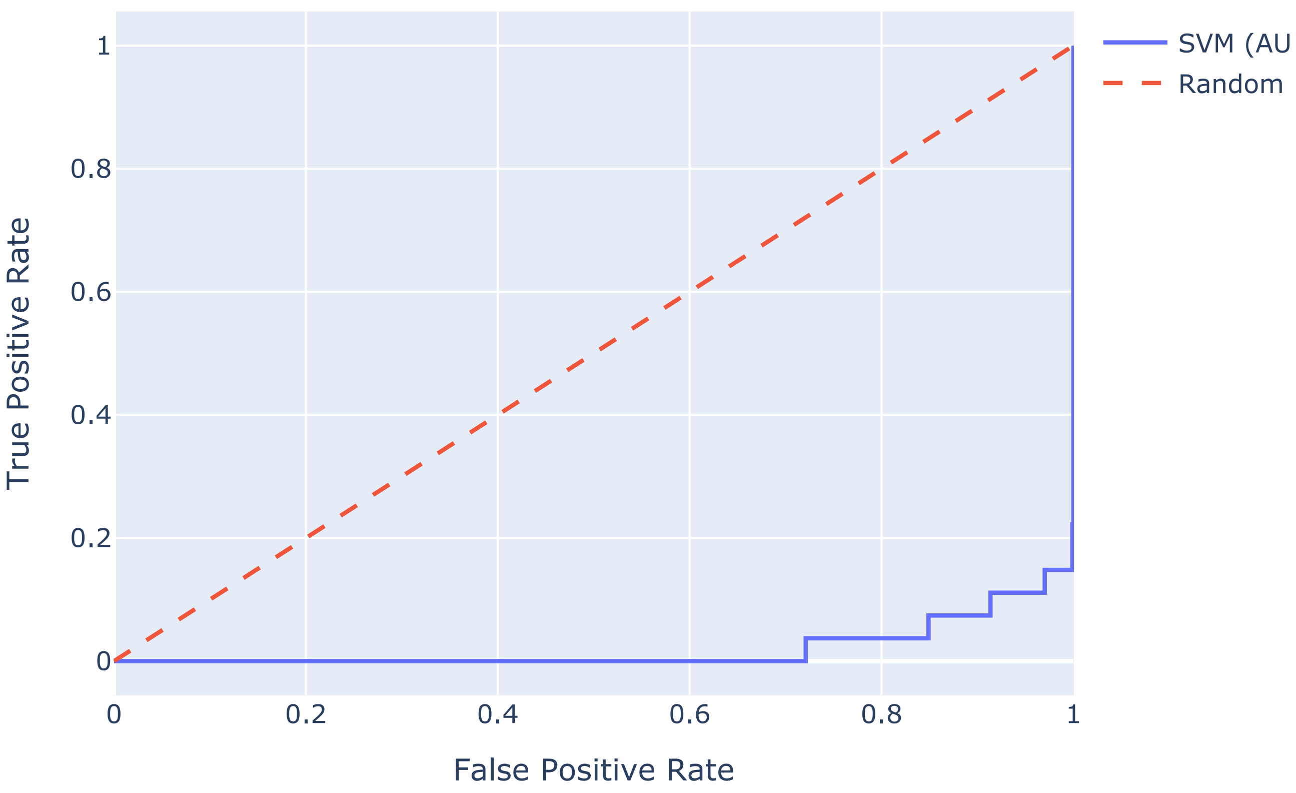


```
🔍 Target: Type_of_Food_Allergy_Cereals_&_Seeds | Model: LogisticRegression
📈 Accuracy: 0.9038
🎯 F1 (0): 0.9491 | F1 (1): 0.0832
📊 Precision: 0.9447 | AUC: 0.5341441682259646
📊 Confusion Matrix:
 [[867   0]
 [ 27   0]]
```



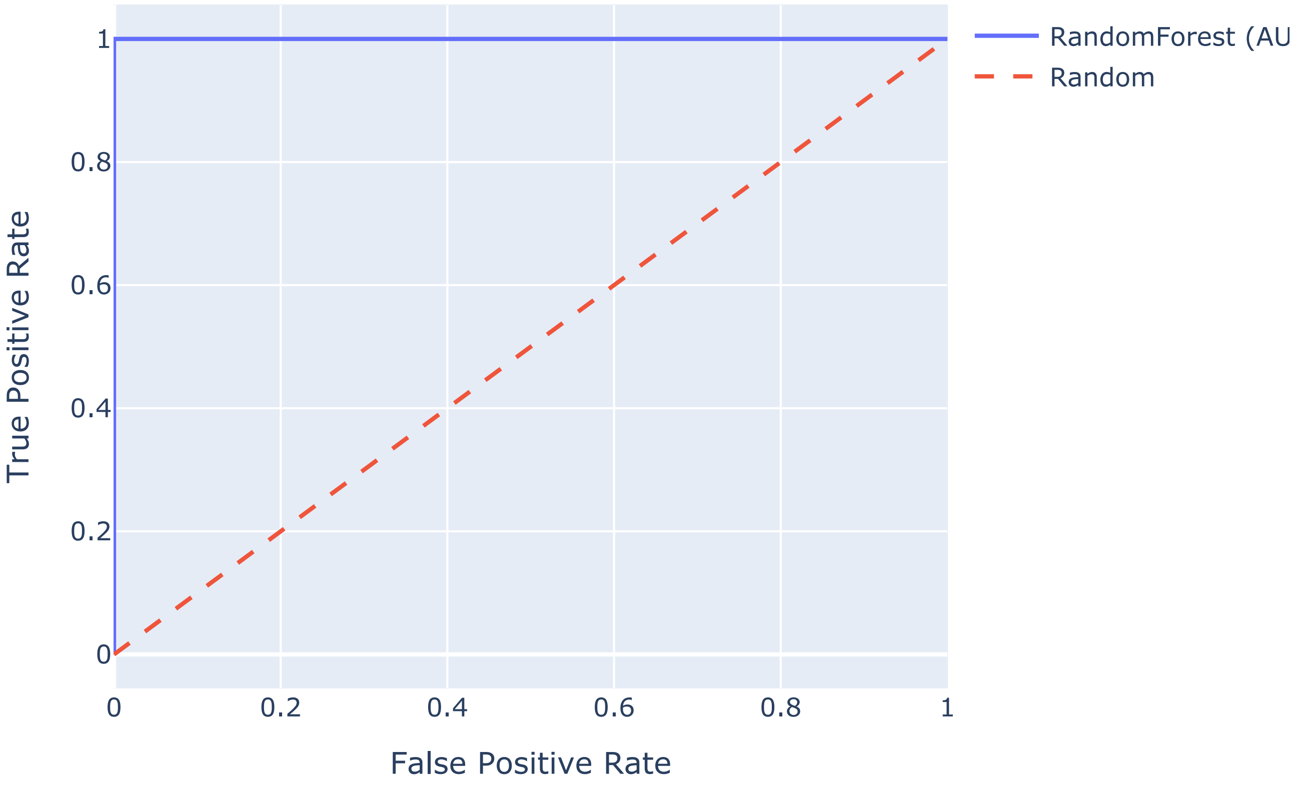
```
🔍 Target: Type_of_Food_Allergy_Cereals_&_Seeds | Model: SVM
📈 Accuracy: 0.7605
🎯 F1 (0): 0.8612 | F1 (1): 0.1009
📊 Precision: 0.9522 | AUC: 0.5535685645549318
📊 Confusion Matrix:
[[867   0]
 [ 27   0]]
```

ROC Curve - Type_of_Food_Allergy_Cereals_&_Seeds - SVM



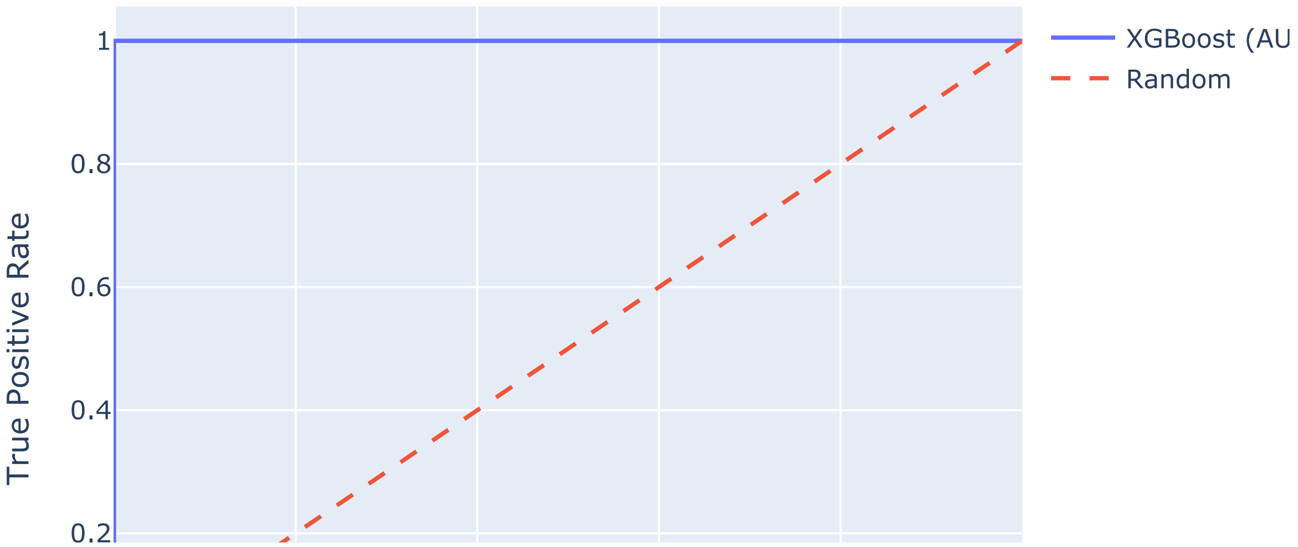
```
🔍 Target: Type_of_Food_Allergy_Egg | Model: RandomForest
📈 Accuracy: 0.9844
🎯 F1 (0): 0.9921 | F1 (1): 0.0667
📊 Precision: 0.9707 | AUC: 0.9022727272727271
📊 Confusion Matrix:
[[880   0]
 [  0  14]]
```

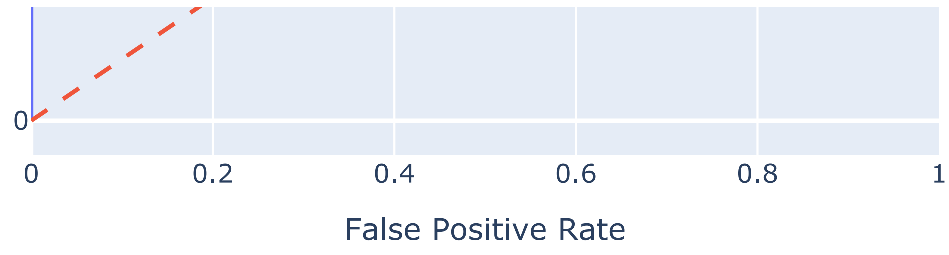
ROC Curve - Type_of_Food_Allergy_Egg - RandomForest



```
🔍 Target: Type_of_Food_Allergy_Egg | Model: XGBoost
📈 Accuracy: 0.9833
🎯 F1 (0): 0.9915 | F1 (1): 0.3500
📊 Precision: 0.9800 | AUC: 0.9335227272727273
📊 Confusion Matrix:
 [[880   0]
 [  0  14]]
```

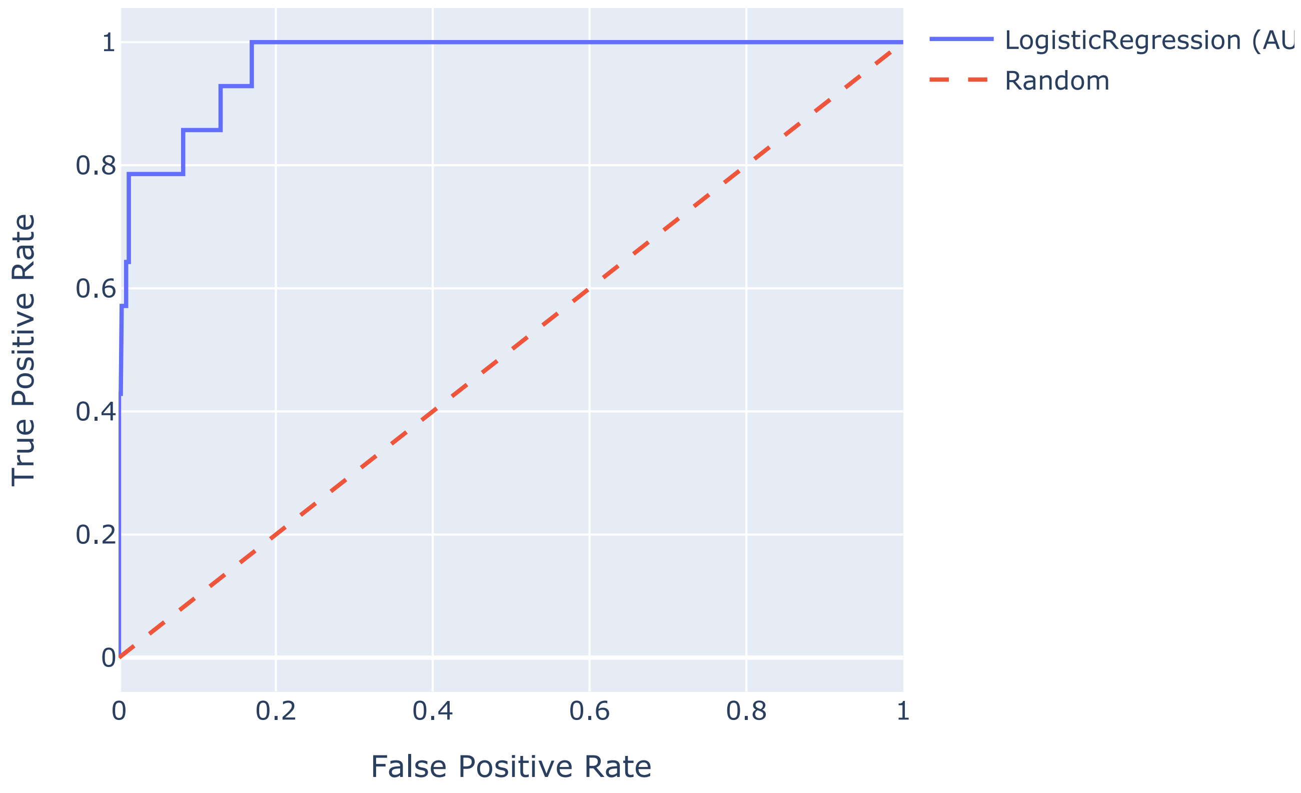
ROC Curve - Type_of_Food_Allergy_Egg - XGBoost



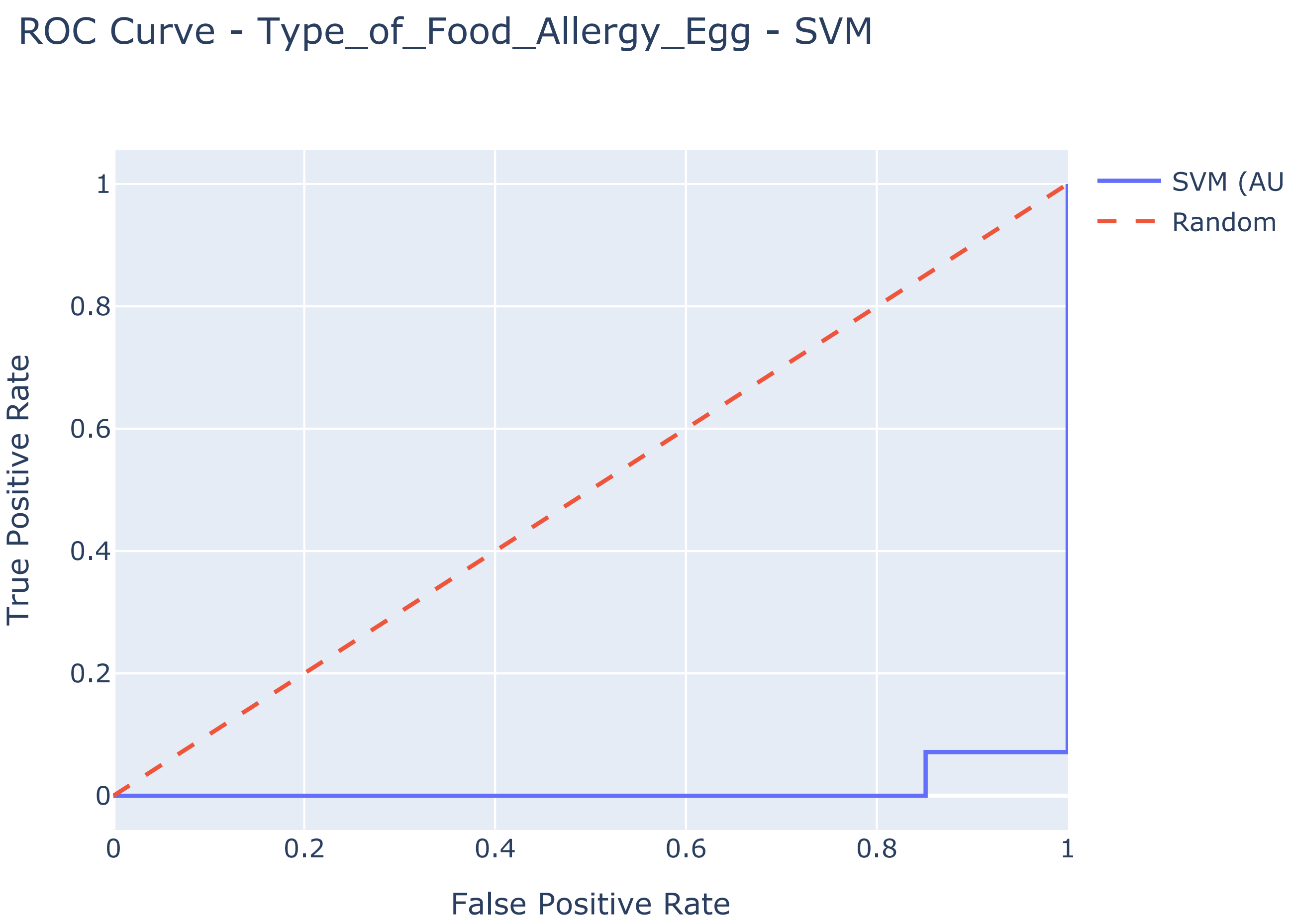


```
🔍 Target: Type_of_Food_Allergy_Egg | Model: LogisticRegression
📈 Accuracy: 0.9519
🎯 F1 (0): 0.9752 | F1 (1): 0.0933
📊 Precision: 0.9726 | AUC: 0.7772727272727273
📊 Confusion Matrix:
 [[880   0]
 [ 12   2]]
```

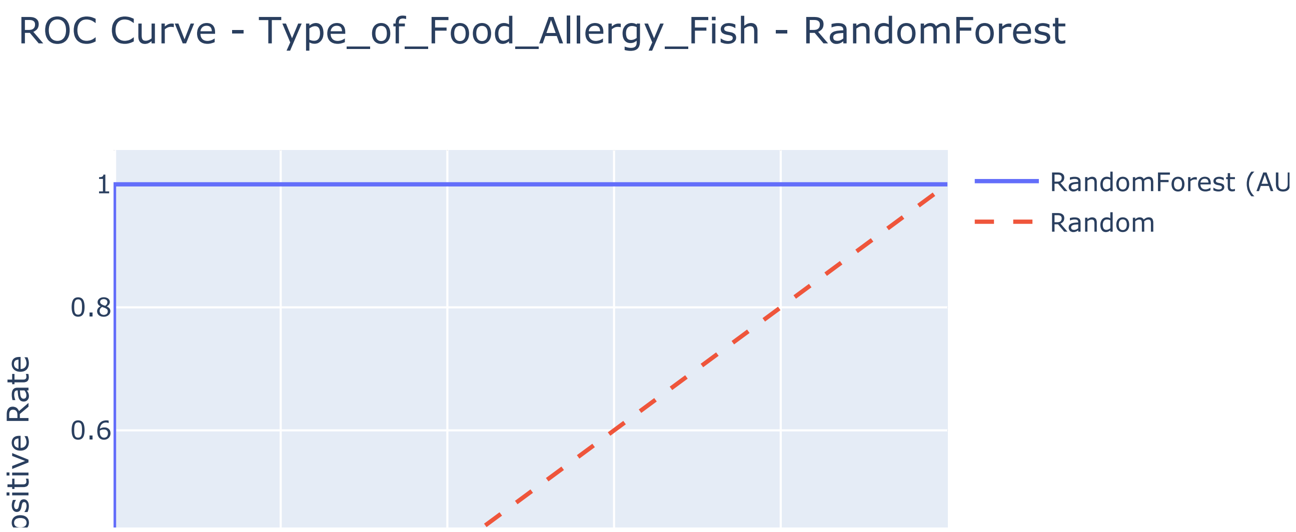
ROC Curve - Type_of_Food_Allergy_Egg - LogisticRegression



```
🔍 Target: Type_of_Food_Allergy_Egg | Model: SVM
📈 Accuracy: 0.8222
🎯 F1 (0): 0.9013 | F1 (1): 0.0520
📊 Precision: 0.9716 | AUC: 0.6448863636363636
📊 Confusion Matrix:
 [[880   0]
 [ 14   0]]
```

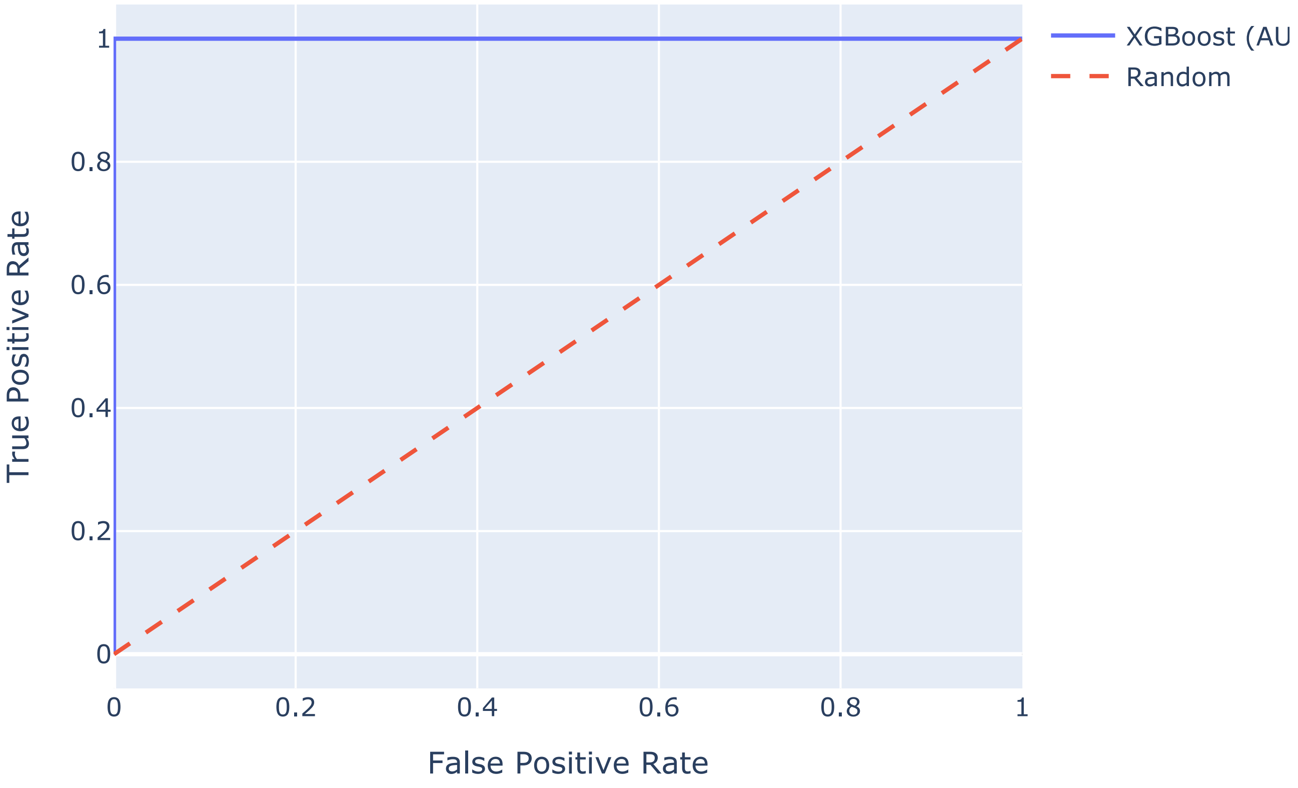


```
🔍 Target: Type_of_Food_Allergy_Fish | Model: RandomForest
📈 Accuracy: 0.9731
🎯 F1 (0): 0.9864 | F1 (1): 0.0000
📊 Precision: 0.9557 | AUC: 0.6779127481713689
📊 Confusion Matrix:
 [[874   0]
 [  0  20]]
```



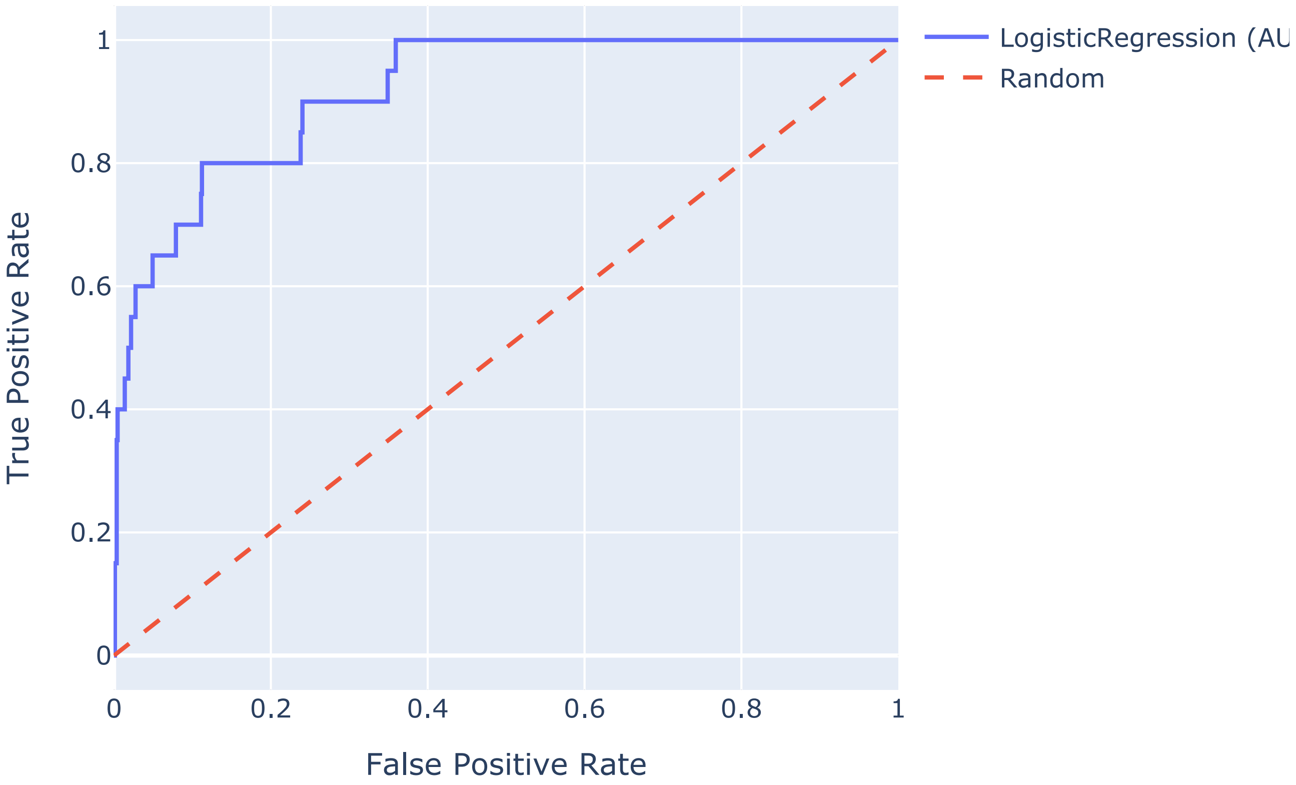
```
🔍 Target: Type_of_Food_Allergy_Fish | Model: XGBoost
📈 Accuracy: 0.9698
🎯 F1 (0): 0.9846 | F1 (1): 0.0667
📊 Precision: 0.9589 | AUC: 0.6398968129571577
📊 Confusion Matrix:
 [[874   0]
 [  0  20]]
```

ROC Curve - Type_of_Food_Allergy_Fish - XGBoost



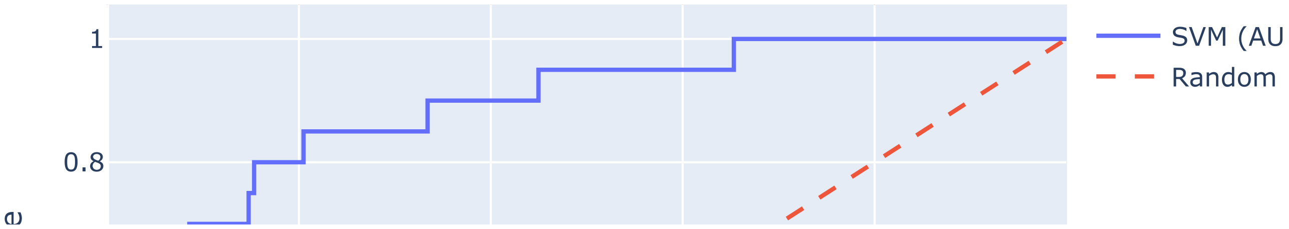
```
🔍 Target: Type_of_Food_Allergy_Fish | Model: LogisticRegression
📈 Accuracy: 0.9239
🎯 F1 (0): 0.9601 | F1 (1): 0.0667
🎯 F1 (0): 0.9601 | F1 (1): 0.0667
📊 Precision: 0.9589 | AUC: 0.550208986415883
📊 Confusion Matrix:
 [[874   0]
 [ 20   0]]
```

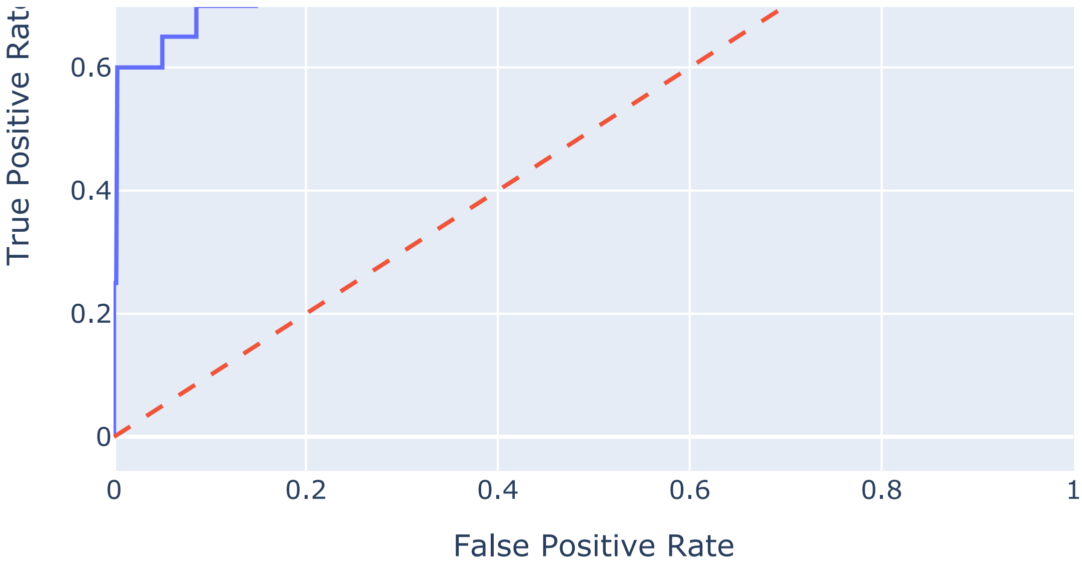
ROC Curve - Type_of_Food_Allergy_Fish - LogisticRegression



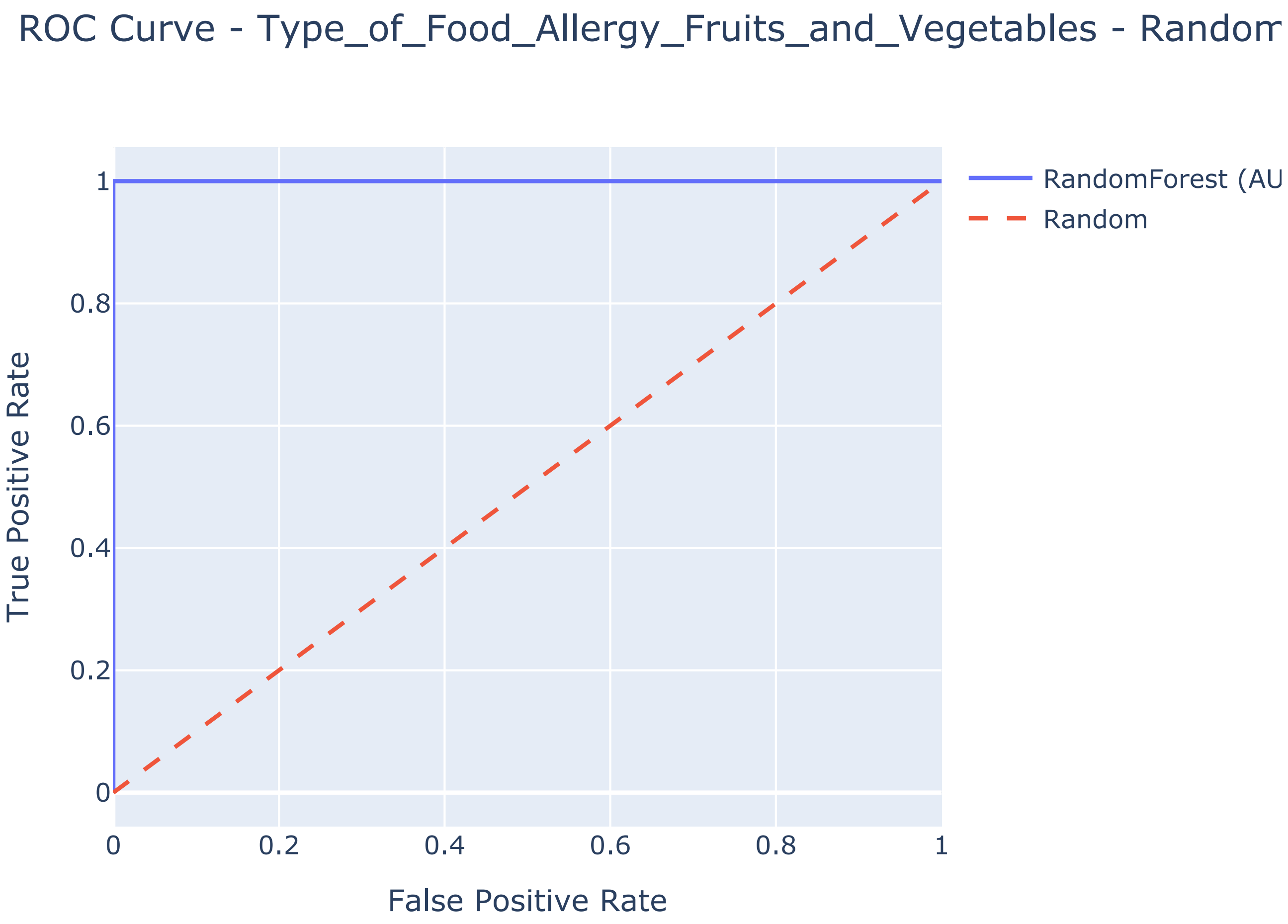
```
🔍 Target: Type_of_Food_Allergy_Fish | Model: SVM
📈 Accuracy: 0.7616
🎯 F1 (0): 0.8624 | F1 (1): 0.0838
📊 Precision: 0.9626 | AUC: 0.6703565830721003
📊 Confusion Matrix:
 [[874   0]
 [ 20   0]]
```

ROC Curve - Type_of_Food_Allergy_Fish - SVM

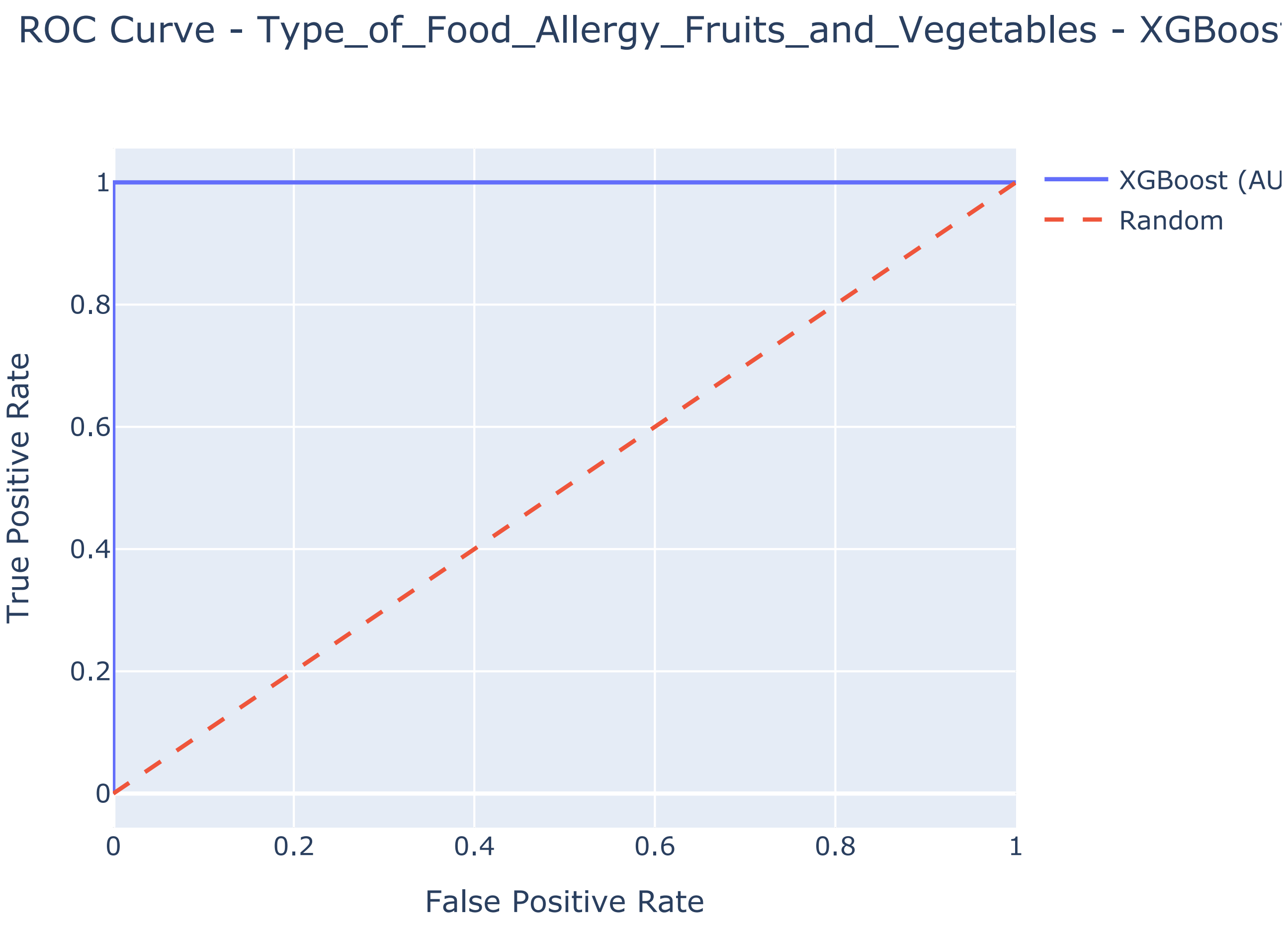




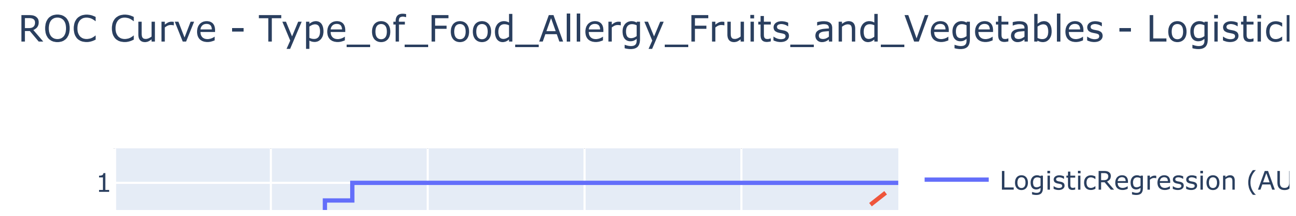
```
🔍 Target: Type_of_Food_Allergy_Fruits_and_Vegetables | Model: RandomForest
📈 Accuracy: 0.9508
🎯 F1 (0): 0.9747 | F1 (1): 0.0733
📊 Precision: 0.9278 | AUC: 0.8438275193798448
📊 Confusion Matrix:
 [[859   0]
 [  0  35]]
```

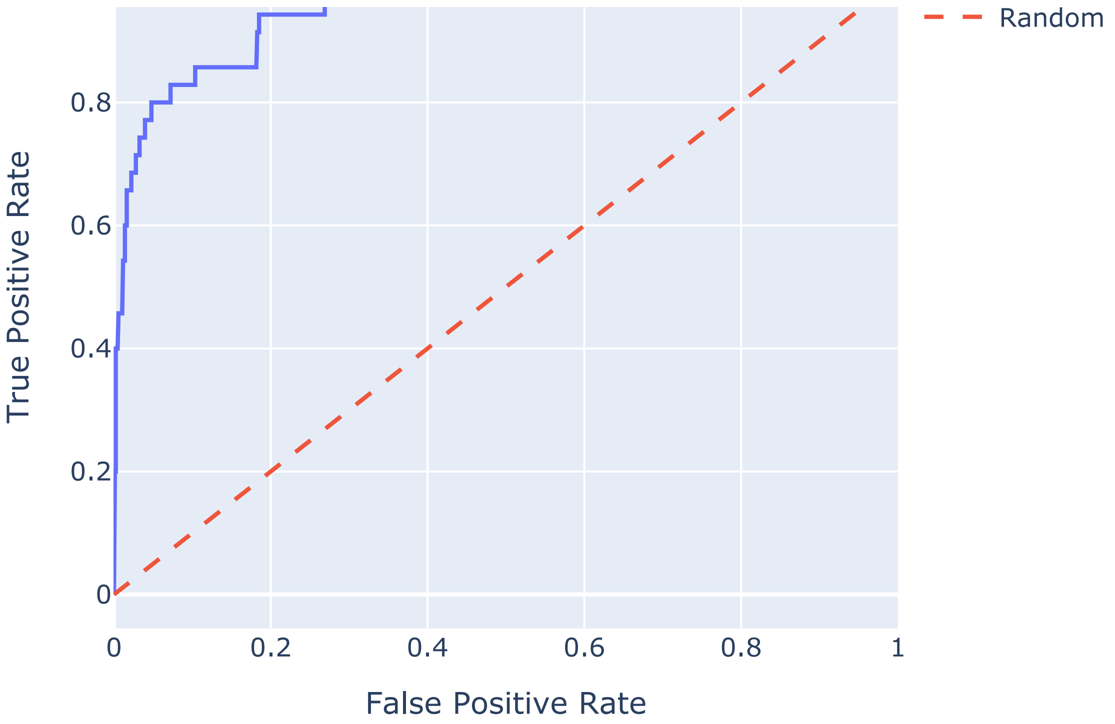


```
🔍 Target: Type_of_Food_Allergy_Fruits_and_Vegetables | Model: XGBoost
📈 Accuracy: 0.9530
🎯 F1 (0): 0.9757 | F1 (1): 0.2305
📊 Precision: 0.9409 | AUC: 0.8612870497036024
📊 Confusion Matrix:
 [[859   0]
 [  0  35]]
```



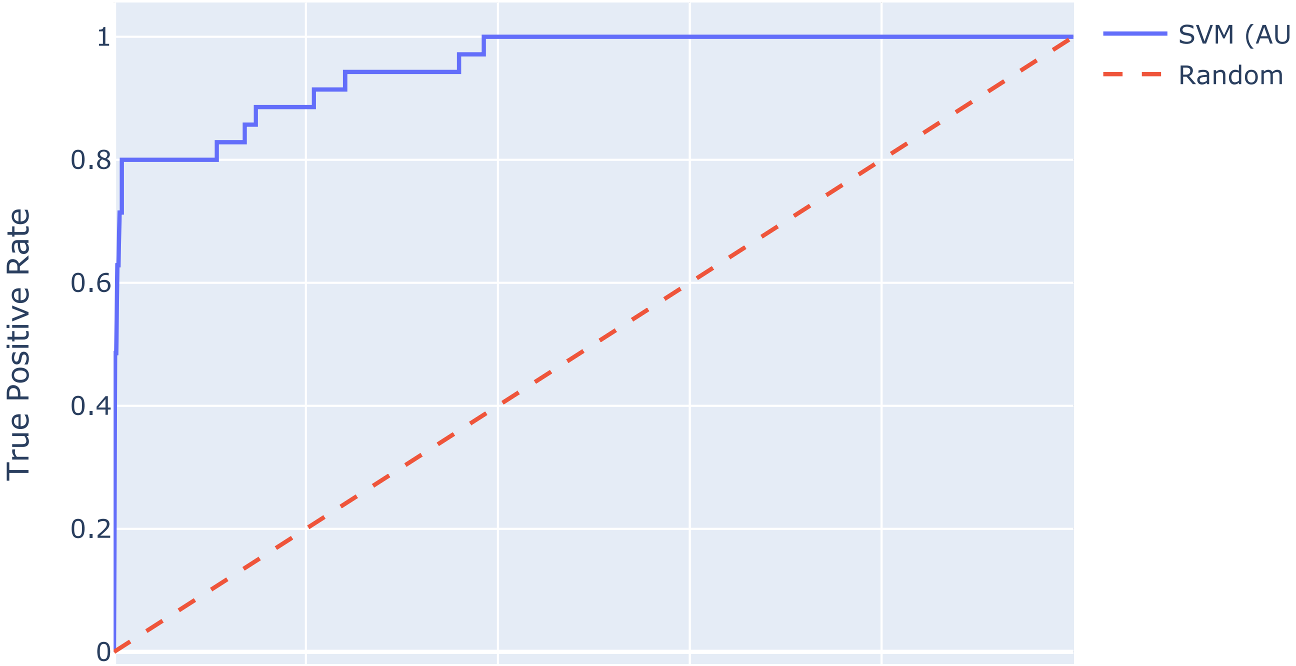
```
🔍 Target: Type_of_Food_Allergy_Fruits_and_Vegetables | Model: LogisticRegr
📈 Accuracy: 0.9162
🎯 F1 (0): 0.9554 | F1 (1): 0.2003
📊 Precision: 0.9395 | AUC: 0.7425227998176015
📊 Confusion Matrix:
 [[858   1]
 [ 28   7]]
```

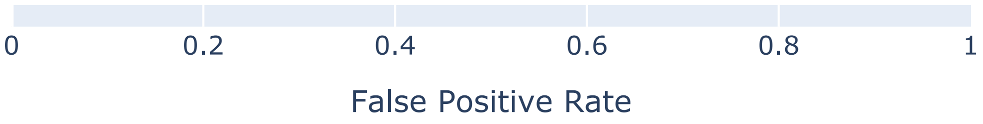




```
🔍 Target: Type_of_Food_Allergy_Fruits_and_Vegetables | Model: SVM
📈 Accuracy: 0.8344
🎯 F1 (0): 0.9069 | F1 (1): 0.2224
📊 Precision: 0.9499 | AUC: 0.8020816233470134
📊 Confusion Matrix:
 [[859   0]
 [ 35   0]]
```

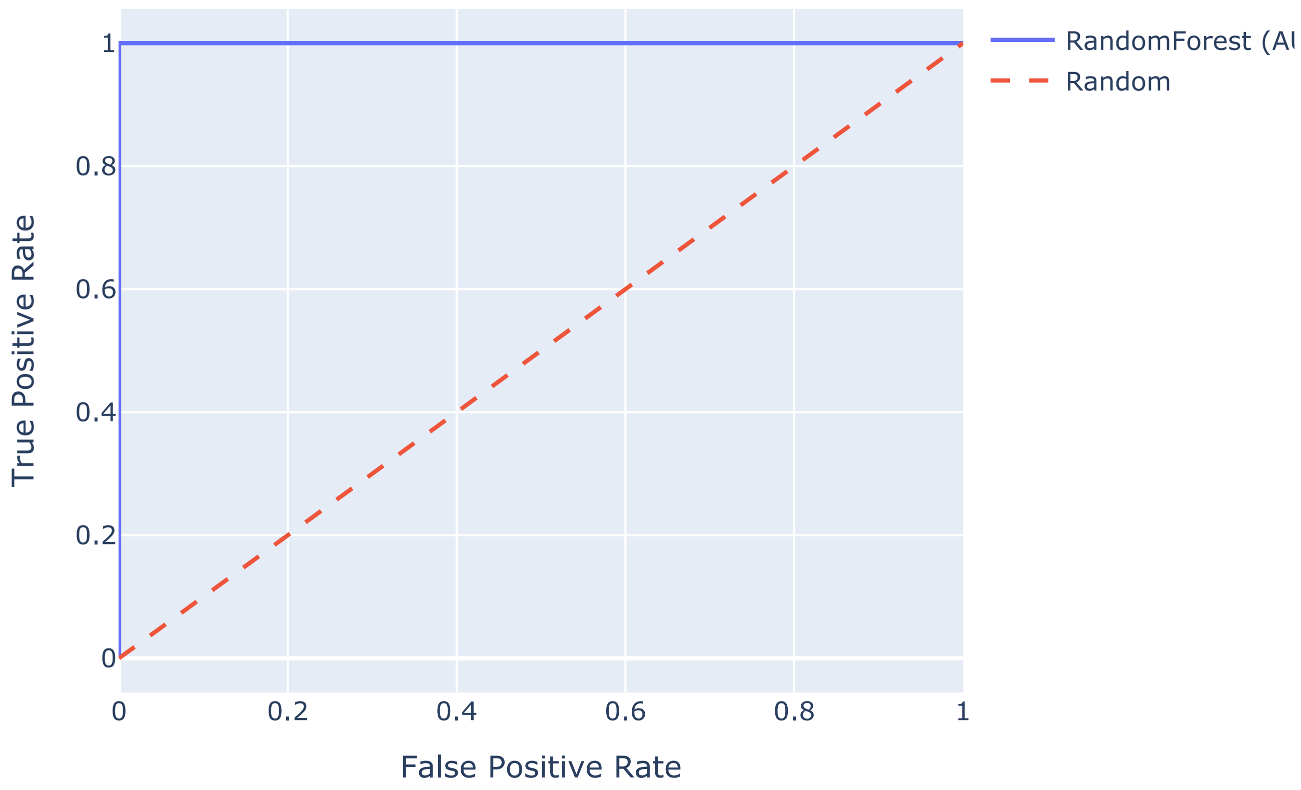
ROC Curve - Type_of_Food_Allergy_Fruits_and_Vegetables - SVM





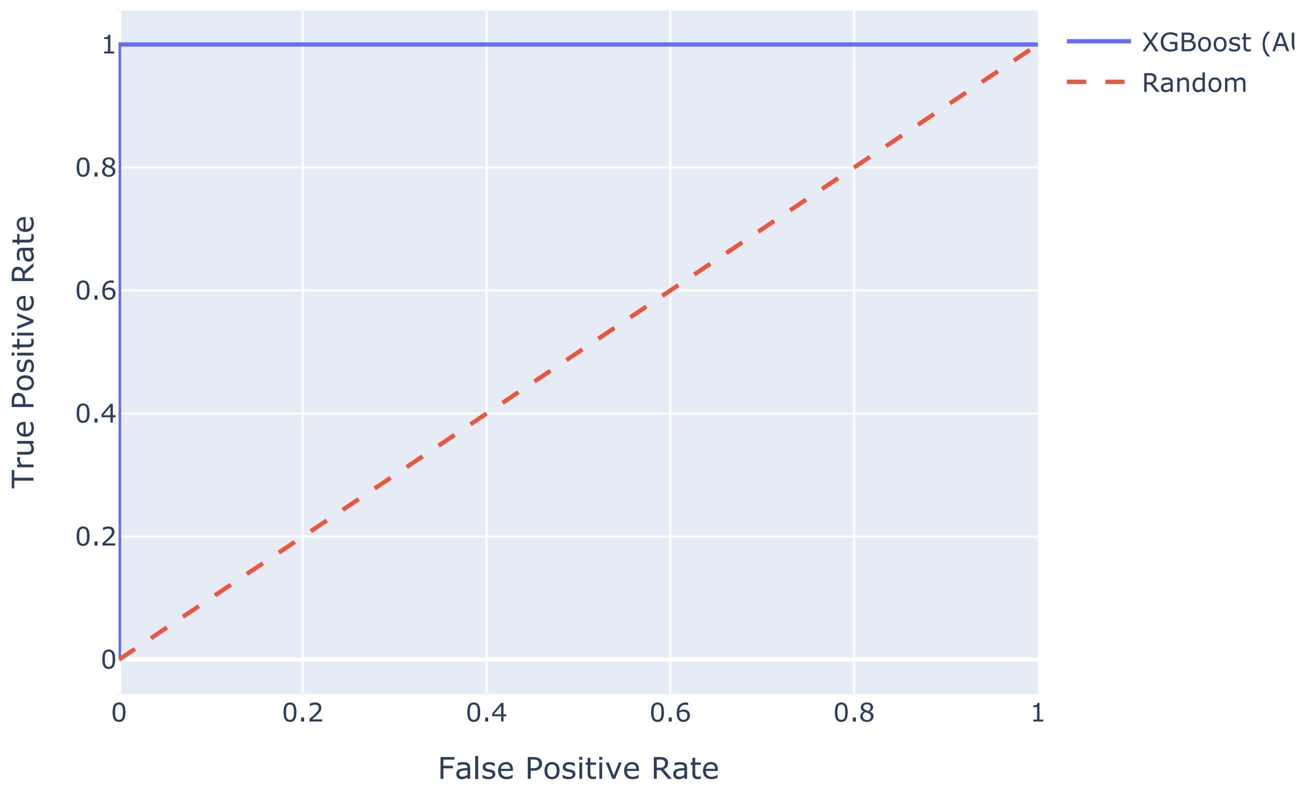
```
🔍 Target: Type_of_Food_Allergy_Mammalian_Milk | Model: RandomForest
📈 Accuracy: 0.9899
🎯 F1 (0): 0.9949 | F1 (1): 0.0000
📊 Precision: 0.9800 | AUC: nan
📊 Confusion Matrix:
 [[885   0]
 [  0   9]]
```

ROC Curve - Type_of_Food_Allergy_Mammalian_Milk - RandomForest

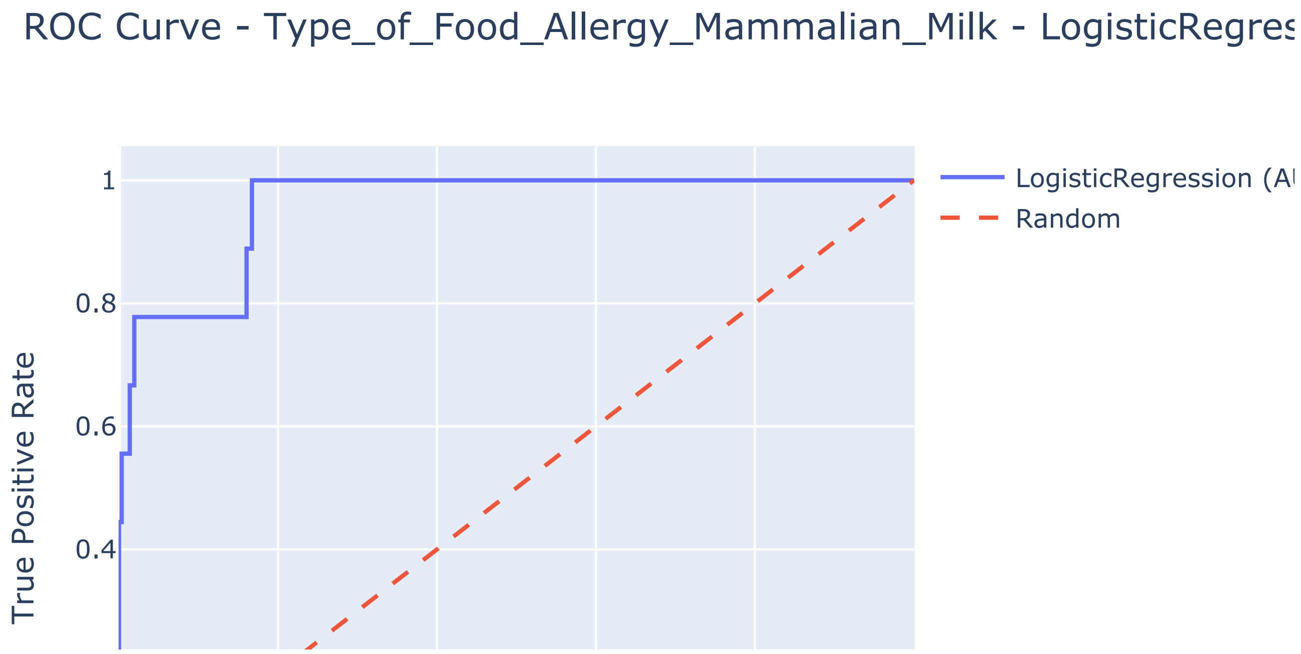


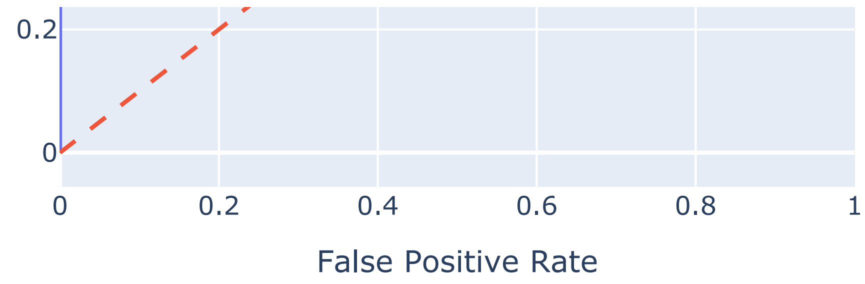
```
🔍 Target: Type_of_Food_Allergy_Mammalian_Milk | Model: XGBoost
📈 Accuracy: 0.9855
🎯 F1 (0): 0.9927 | F1 (1): 0.0000
📊 Precision: 0.9799 | AUC: nan
📊 Confusion Matrix:
 [[885   0]
 [  0   9]]
```

ROC Curve - Type_of_Food_Allergy_Mammalian_Milk - XGBoost

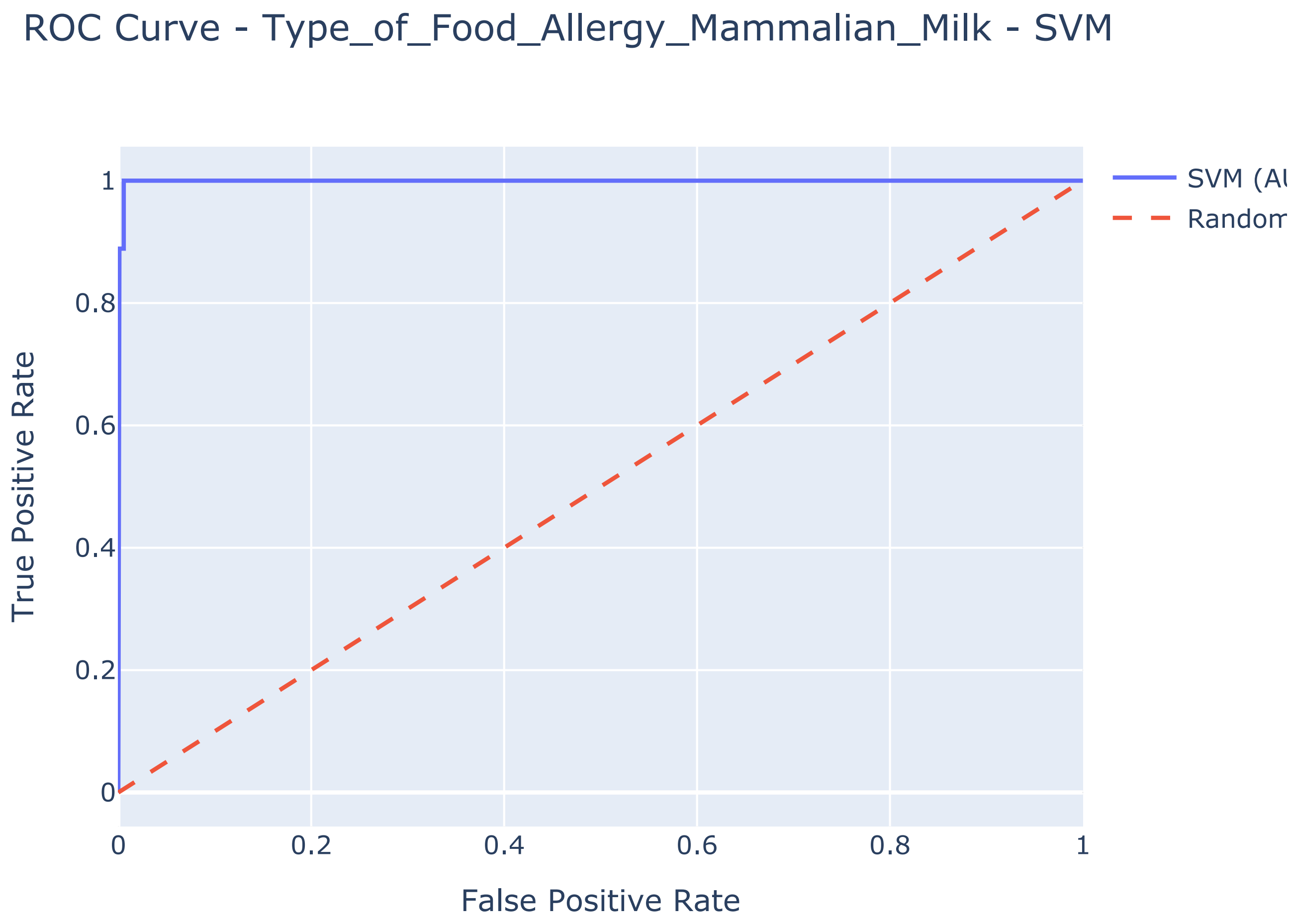


```
🔍 Target: Type_of_Food_Allergy_Mammalian_Milk | Model: LogisticRegression
📈 Accuracy: 0.9697
🎯 F1 (0): 0.9846 | F1 (1): 0.0000
📊 Precision: 0.9798 | AUC: nan
📊 Confusion Matrix:
 [[885   0]
 [  9   0]]
```

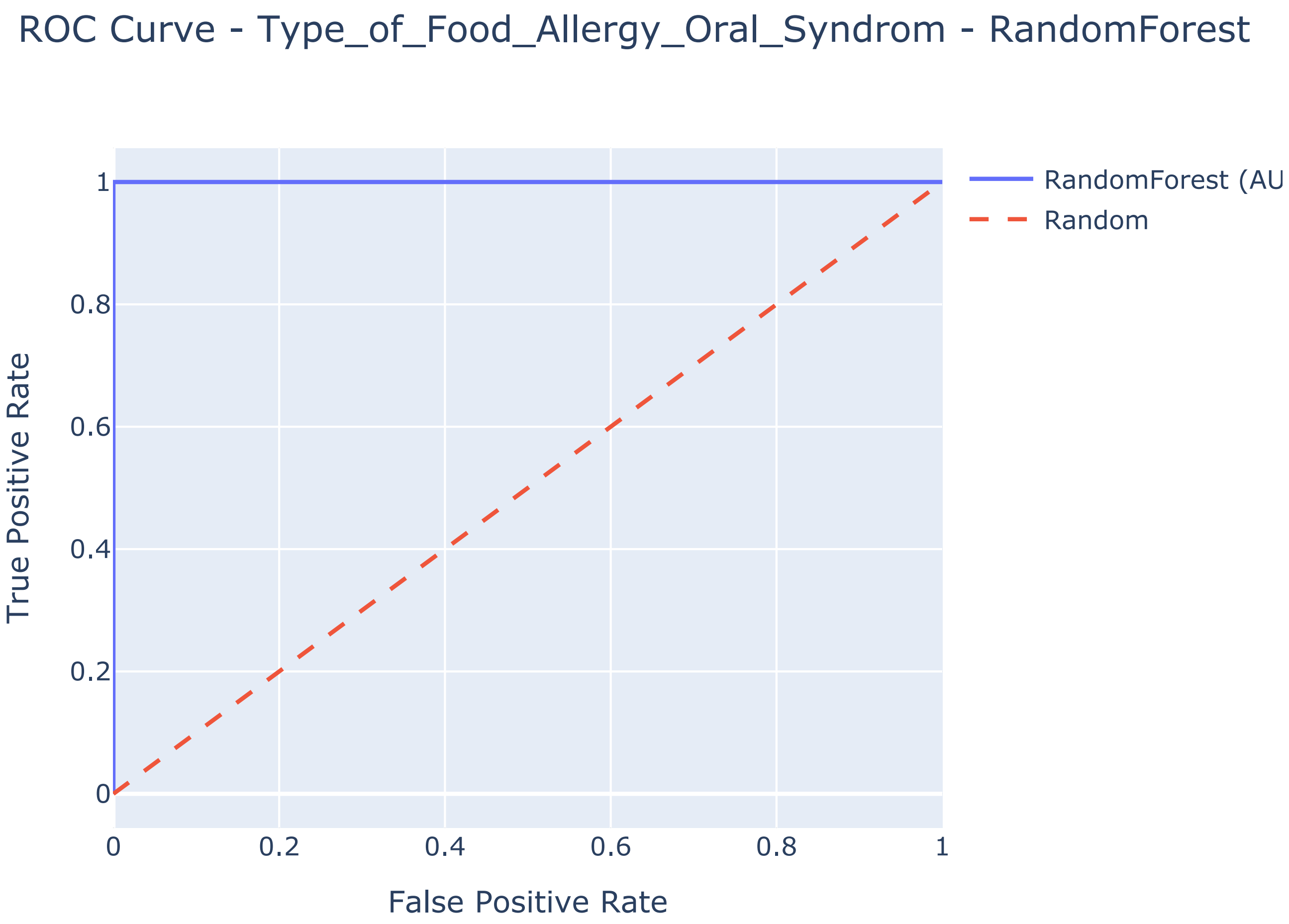




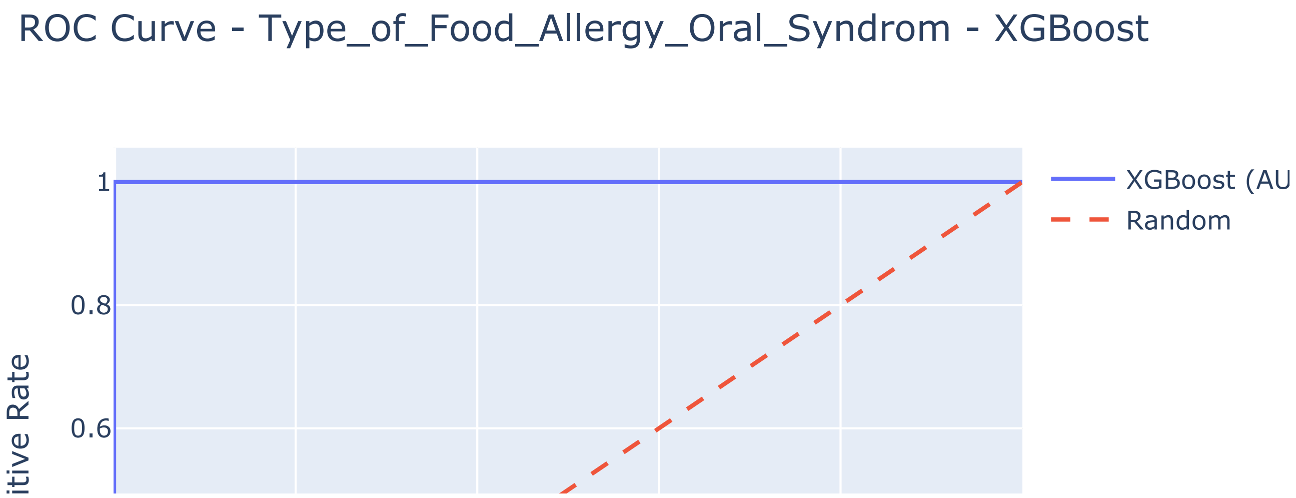
```
🔍 Target: Type_of_Food_Allergy_Mammalian_Milk | Model: SVM
📈 Accuracy: 0.8232
🎯 F1 (0): 0.9022 | F1 (1): 0.0000
📊 Precision: 0.9780 | AUC: nan
📊 Confusion Matrix:
 [[885   0]
 [  9   0]]
```


ROC Curve - Type_of_Food_Allergy_Mammalian_Milk - SVM

```
🔍 Target: Type_of_Food_Allergy_Oral_Syndrom | Model: RandomForest
📈 Accuracy: 0.9732
🎯 F1 (0): 0.9849 | F1 (1): 0.8815
📊 Precision: 0.9742 | AUC: 0.9989073426573427
📊 Confusion Matrix:
 [[777   0]
 [   0 117]]
```

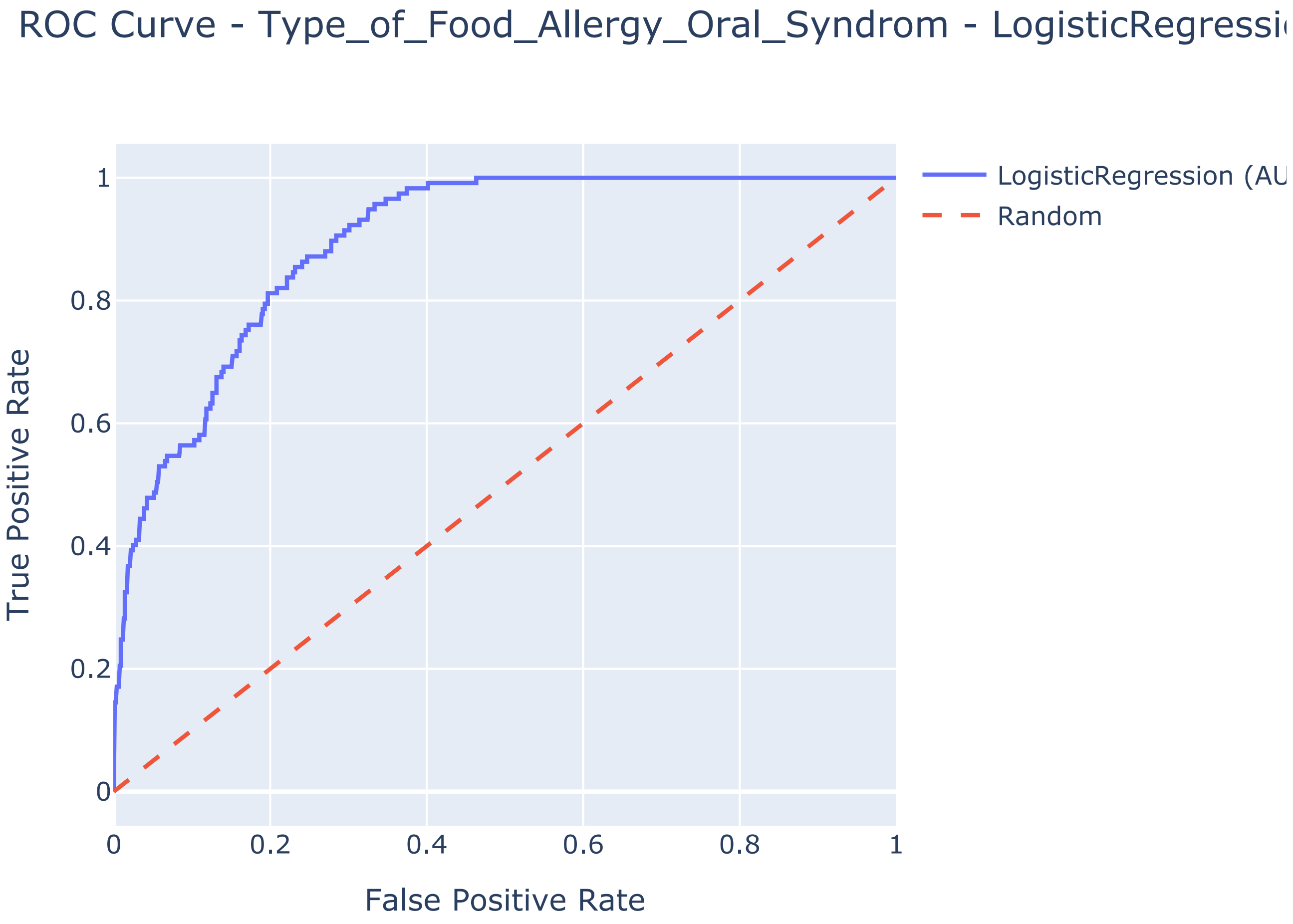


```
🔍 Target: Type_of_Food_Allergy_Oral_Syndrom | Model: XGBoost
📈 Accuracy: 0.9989
🎯 F1 (0): 0.9994 | F1 (1): 0.9960
📊 Precision: 0.9990 | AUC: 0.999465811965812
📊 Confusion Matrix:
 [[777   0]
 [   0 117]]
```

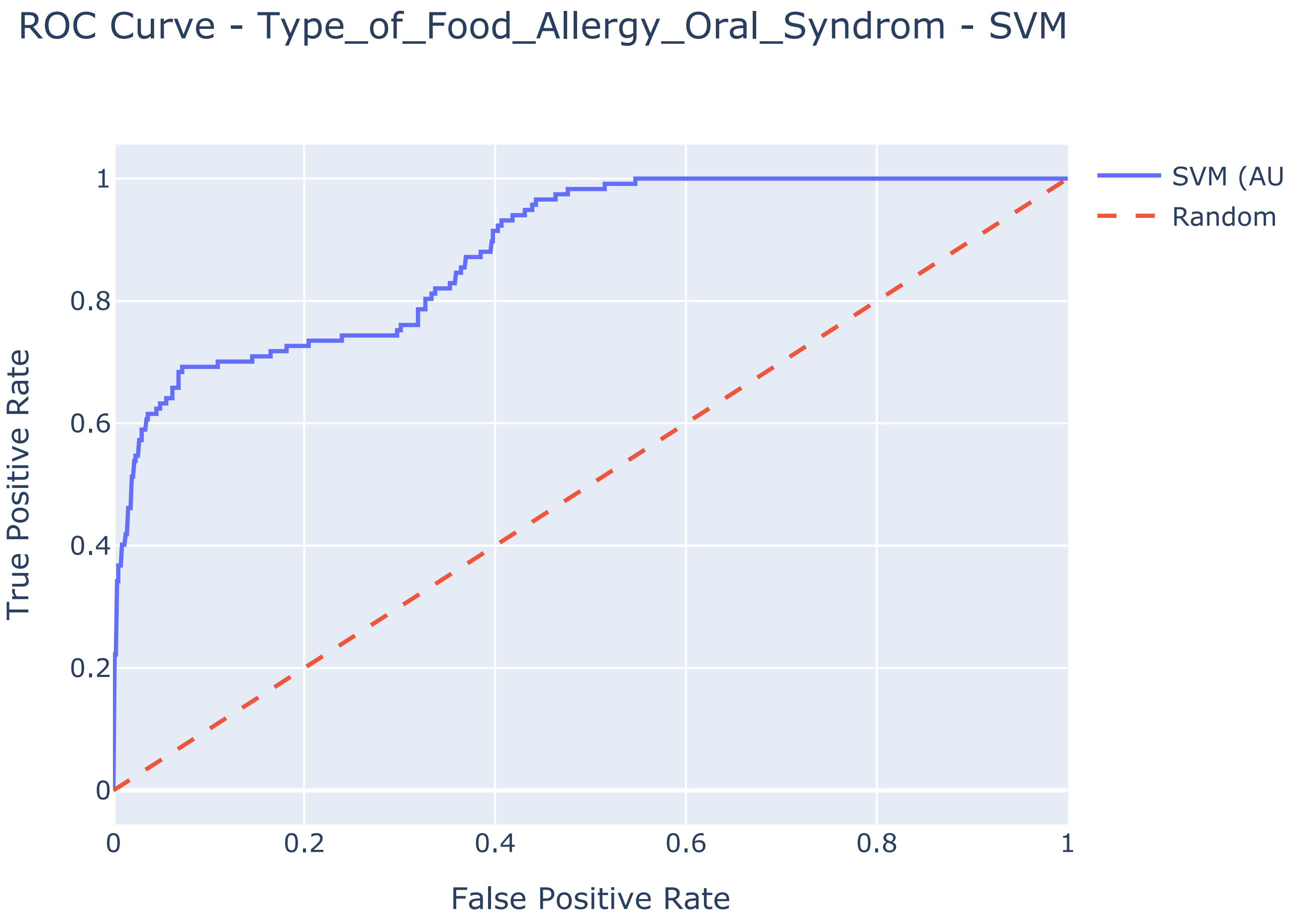


True Pos
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
False Positive Rate

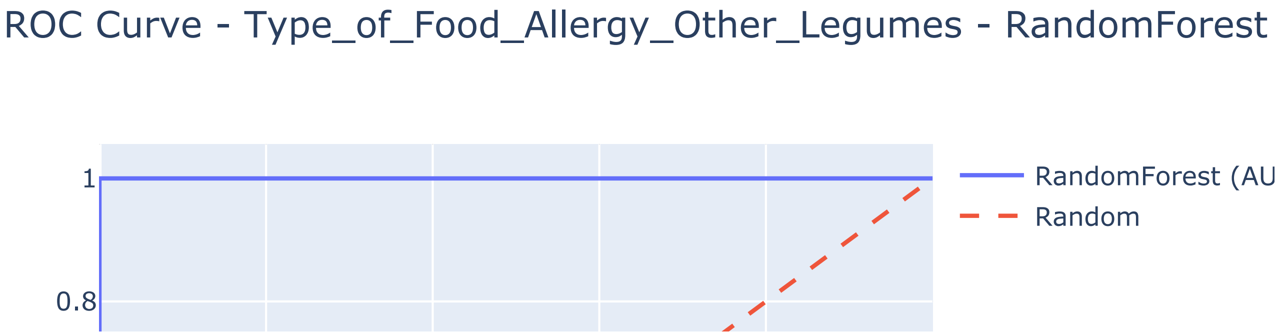
```
🔍 Target: Type_of_Food_Allergy_Oral_Syndrom | Model: LogisticRegression
📈 Accuracy: 0.8379
🎯 F1 (0): 0.9058 | F1 (1): 0.4118
📊 Precision: 0.8469 | AUC: 0.8055735930735931
📊 Confusion Matrix:
 [[764  13]
 [ 77  40]]
```

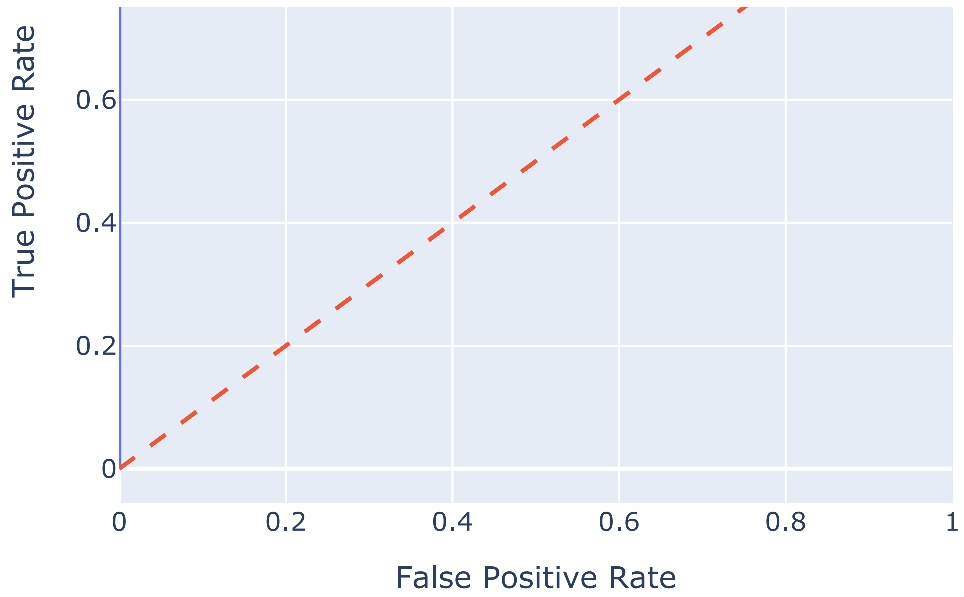


```
🔍 Target: Type_of_Food_Allergy_Oral_Syndrom | Model: SVM
📈 Accuracy: 0.6700
📈 Accuracy: 0.6700
🎯 F1 (0): 0.7741 | F1 (1): 0.3789
📊 Precision: 0.8590 | AUC: 0.7192404817404817
📊 Confusion Matrix:
 [[777   0]
 [116   1]]
```

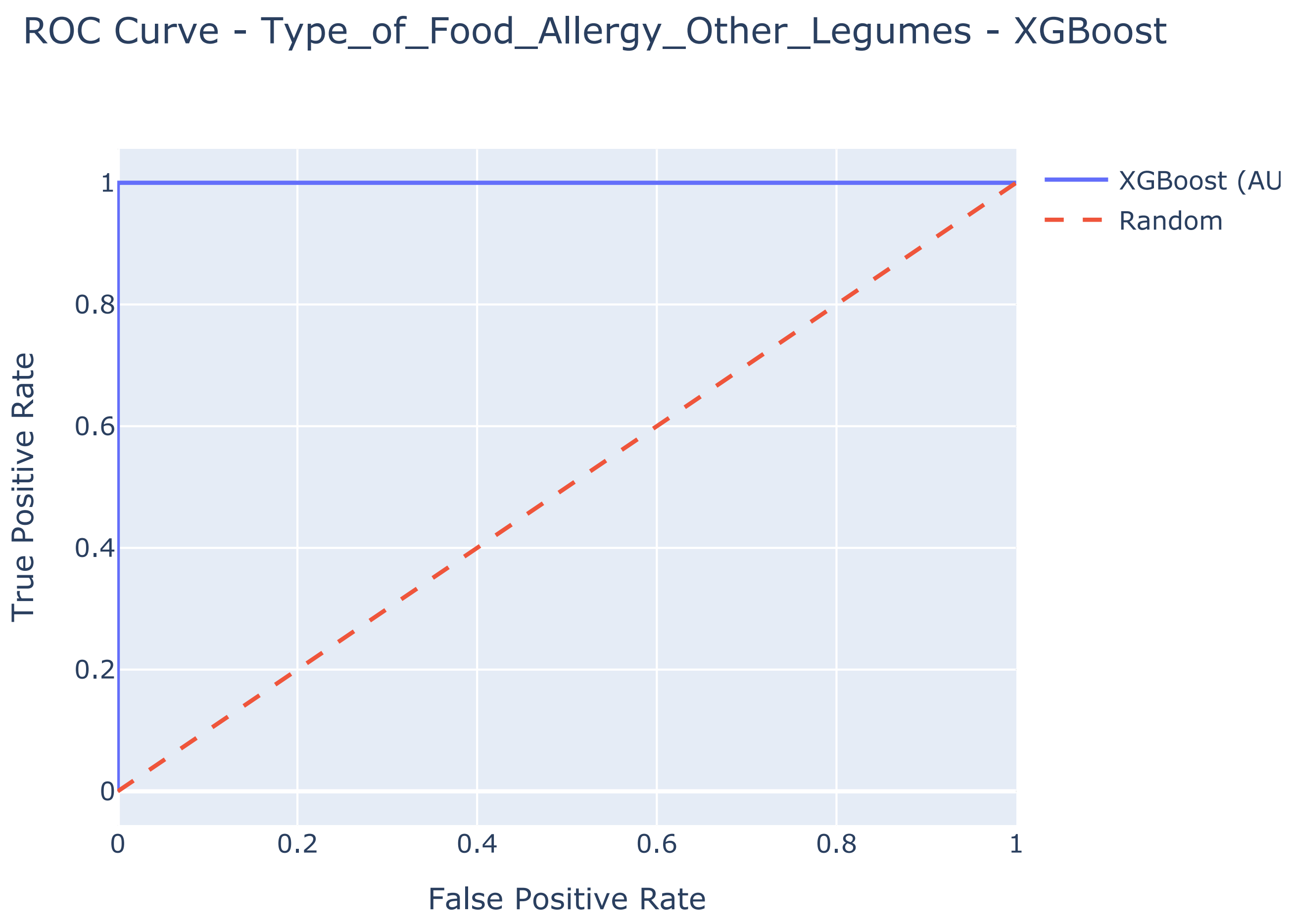

ROC Curve - Type_of_Food_Allergy_Oral_Syndrom - SVM

```
🔍 Target: Type_of_Food_Allergy_Other_Legumes | Model: RandomForest
📈 Accuracy: 0.9799
🎯 F1 (0): 0.9898 | F1 (1): 0.0000
📊 Precision: 0.9646 | AUC: 0.7299536311389759
📊 Confusion Matrix:
 [[878   0]
 [  0  16]]
```

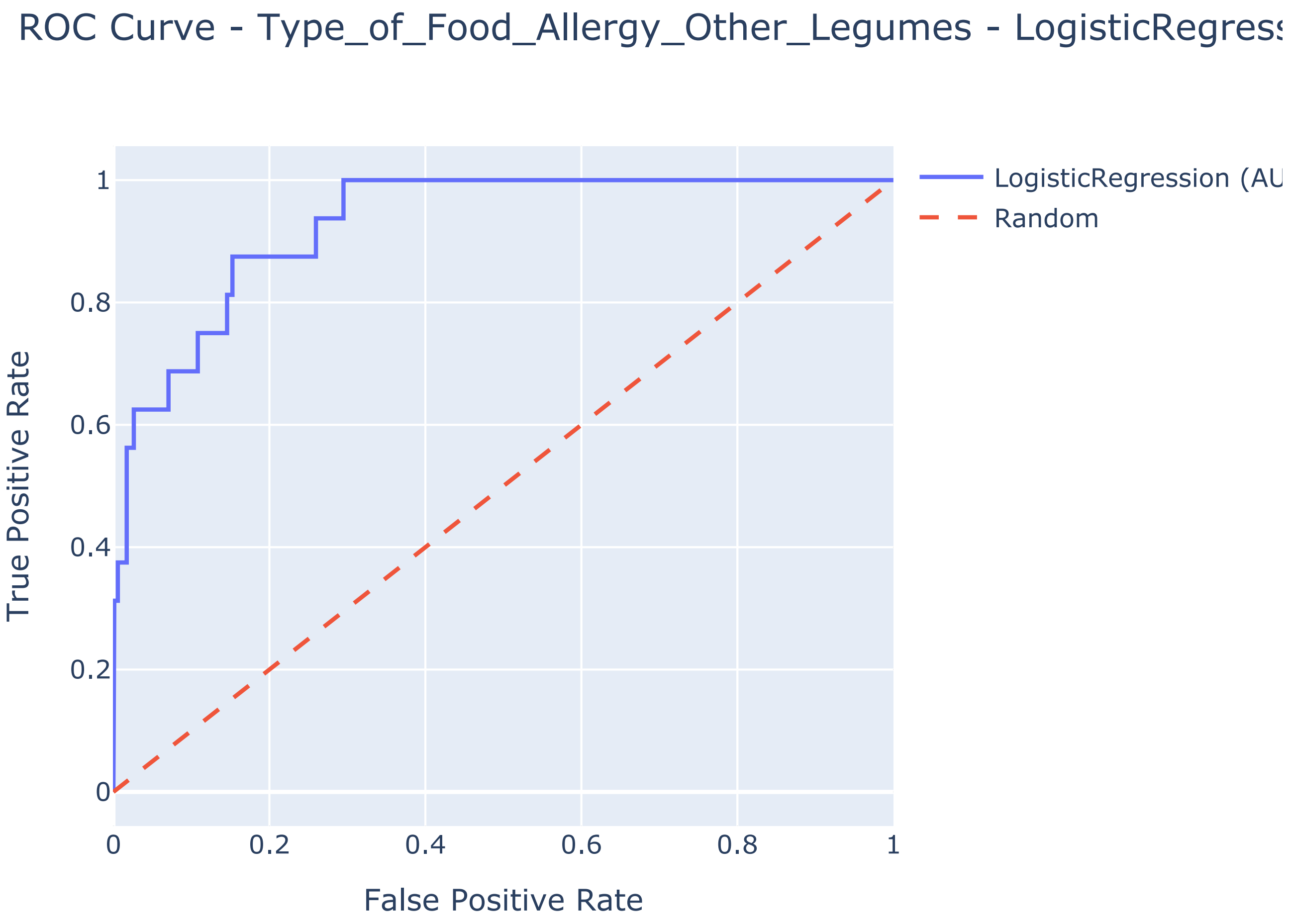

ROC Curve - Type_of_Food_Allergy_Other_Legumes - RandomForest



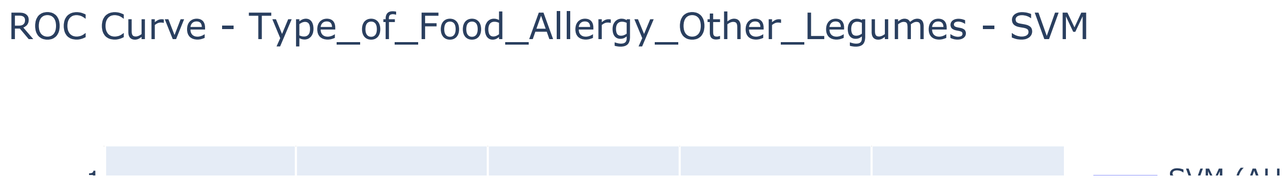
```
🔍 Target: Type_of_Food_Allergy_Other_Legumes | Model: XGBoost
📈 Accuracy: 0.9788
🎯 F1 (0): 0.9892 | F1 (1): 0.0667
📊 Precision: 0.9662 | AUC: 0.6641588296760711
📊 Confusion Matrix:
 [[878   0]
 [  0  16]]
```

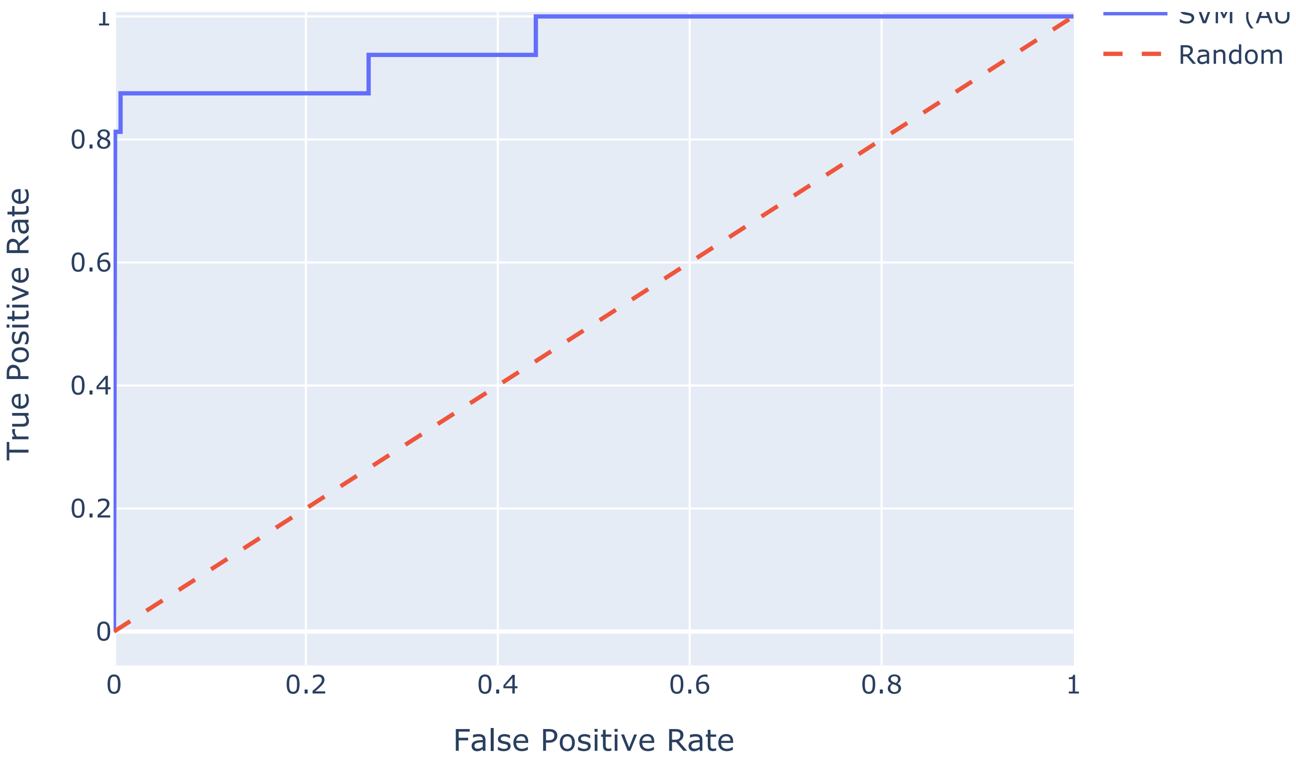


```
🔍 Target: Type_of_Food_Allergy_Other_Legumes | Model: LogisticRegression
📈 Accuracy: 0.9474
🎯 F1 (0): 0.9728 | F1 (1): 0.0686
📊 Precision: 0.9667 | AUC: 0.5197165621734587
📊 Confusion Matrix:
 [[878   0]
 [ 16   0]]
```


ROC Curve - Type_of_Food_Allergy_Other_Legumes - LogisticRegress

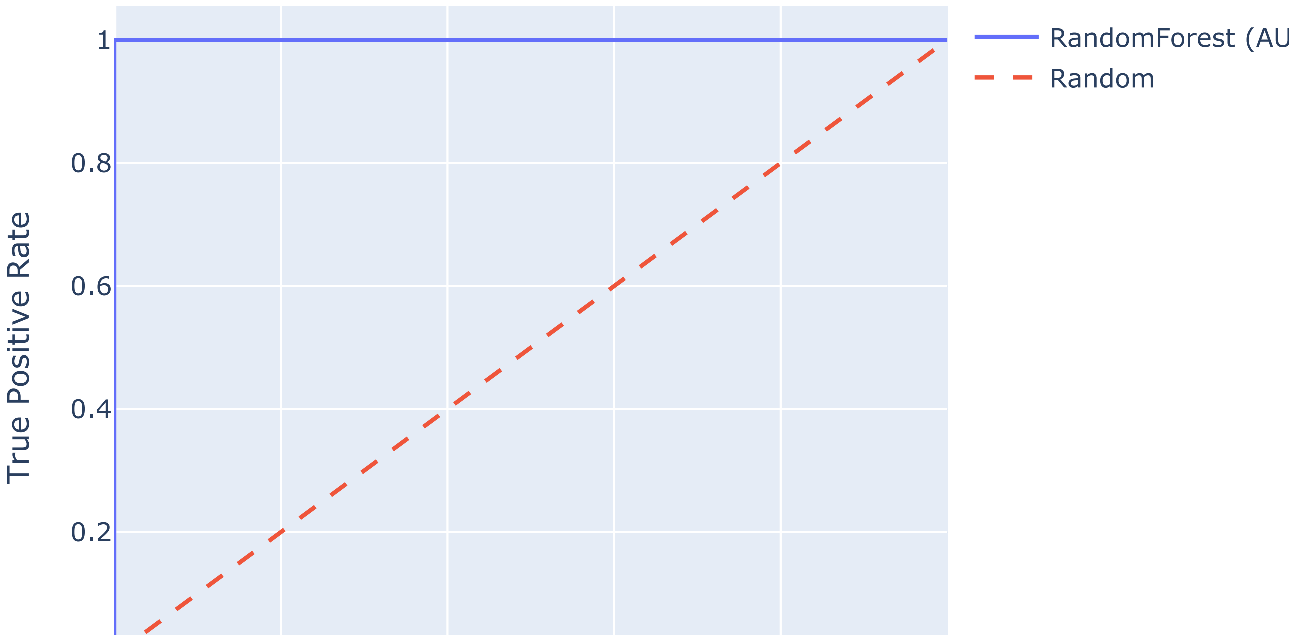
```
🔍 Target: Type_of_Food_Allergy_Other_Legumes | Model: SVM
📈 Accuracy: 0.7806
🎯 F1 (0): 0.8747 | F1 (1): 0.0473
📊 Precision: 0.9677 | AUC: 0.5754049111807733
📊 Confusion Matrix:
 [[878   0]
 [ 16   0]]
```

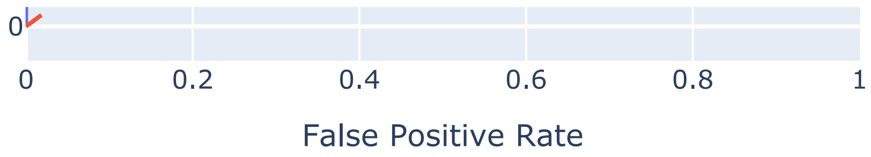

ROC Curve - Type_of_Food_Allergy_Other_Legumes - SVM



```
🔍 Target: Type_of_Food_Allergy_Peanut | Model: RandomForest
📈 Accuracy: 0.9127
🎯 F1 (0): 0.9536 | F1 (1): 0.2453
📊 Precision: 0.8940 | AUC: 0.8190126359094917
📊 Confusion Matrix:
 [[826   0]
 [  0  68]]
```

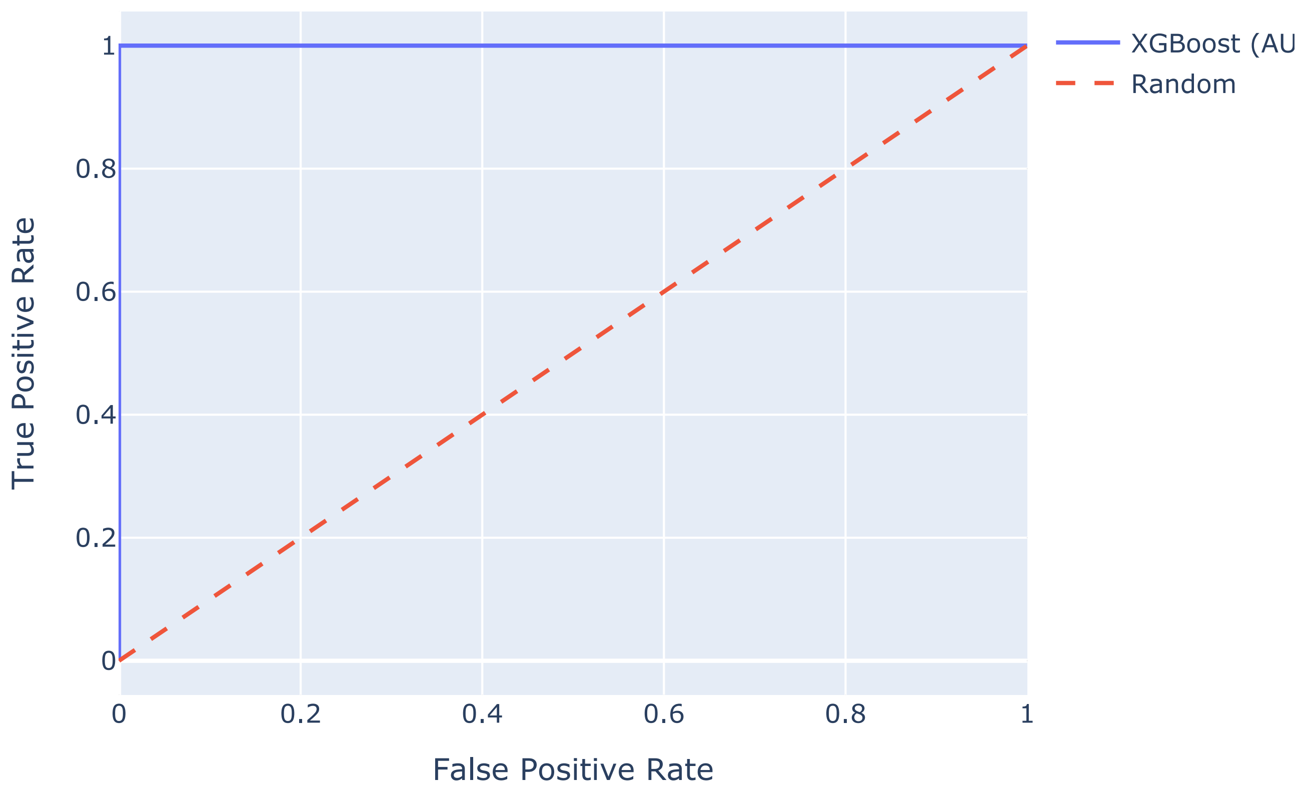
ROC Curve - Type_of_Food_Allergy_Peanut - RandomForest





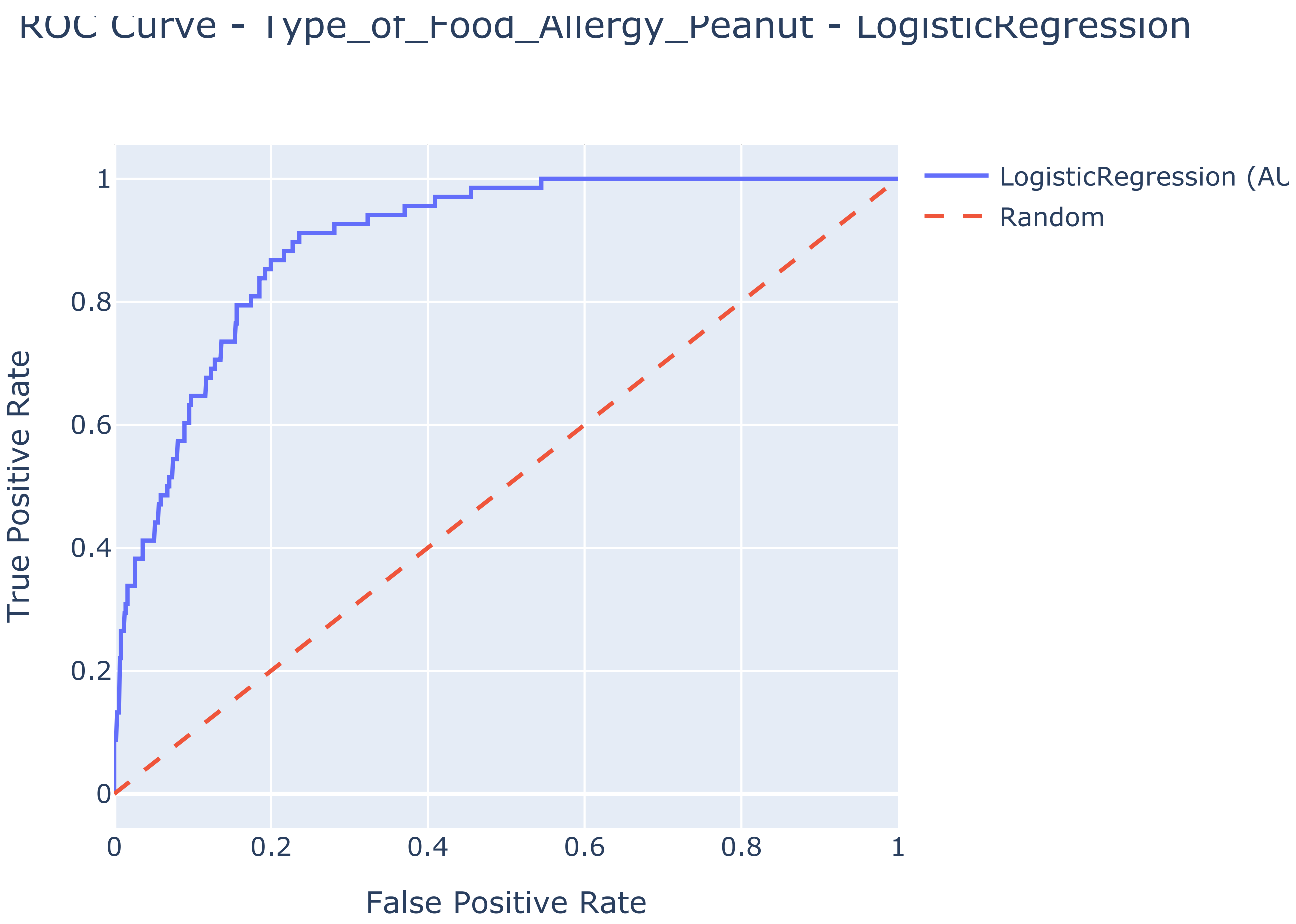
```
🔍 Target: Type_of_Food_Allergy_Peanut | Model: XGBoost
📈 Accuracy: 0.9072
🎯 F1 (0): 0.9502 | F1 (1): 0.3065
📊 Precision: 0.8972 | AUC: 0.8572869876719421
📊 Confusion Matrix:
 [[826   0]
 [  0  68]]
```

ROC Curve - Type_of_Food_Allergy_Peanut - XGBoost



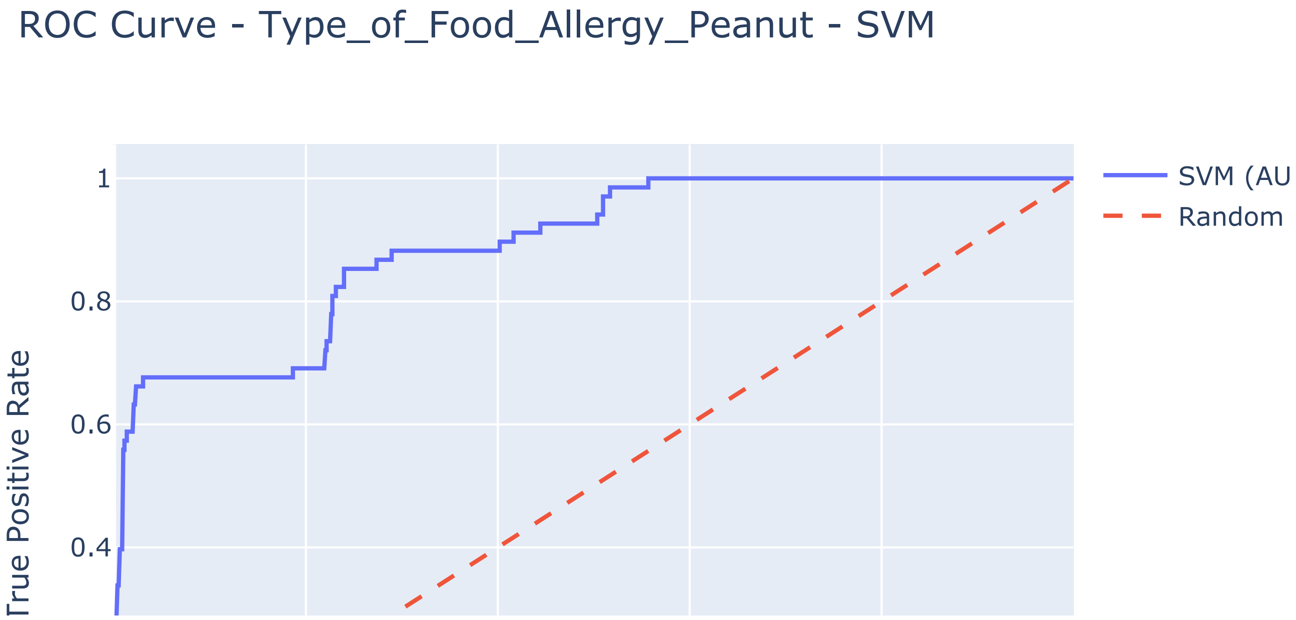
```
🔍 Target: Type_of_Food_Allergy_Peanut | Model: LogisticRegression
📈 Accuracy: 0.8657
🎯 F1 (0): 0.9260 | F1 (1): 0.2537
📊 Precision: 0.8876 | AUC: 0.6785497390257895
📊 Confusion Matrix:
 [[821   5]
 [ 55  13]]
```

ROC Curve - Type_of_Food_Allergy_Peanut - LogisticRegression

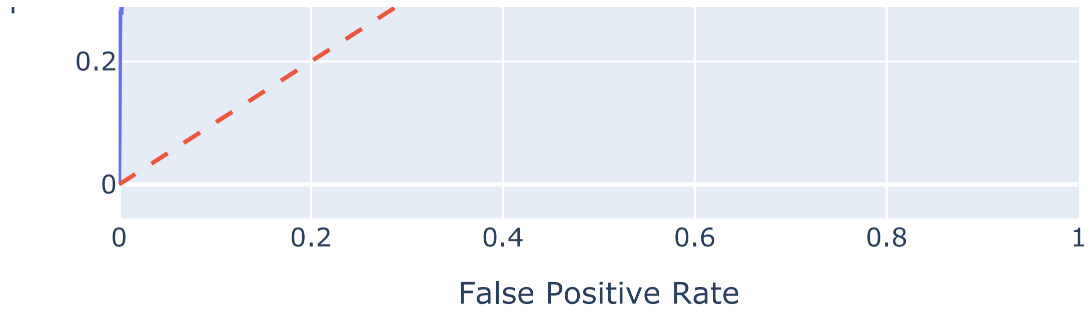


ROC Curve - Type_of_Food_Allergy_Peanut - LogisticRegression

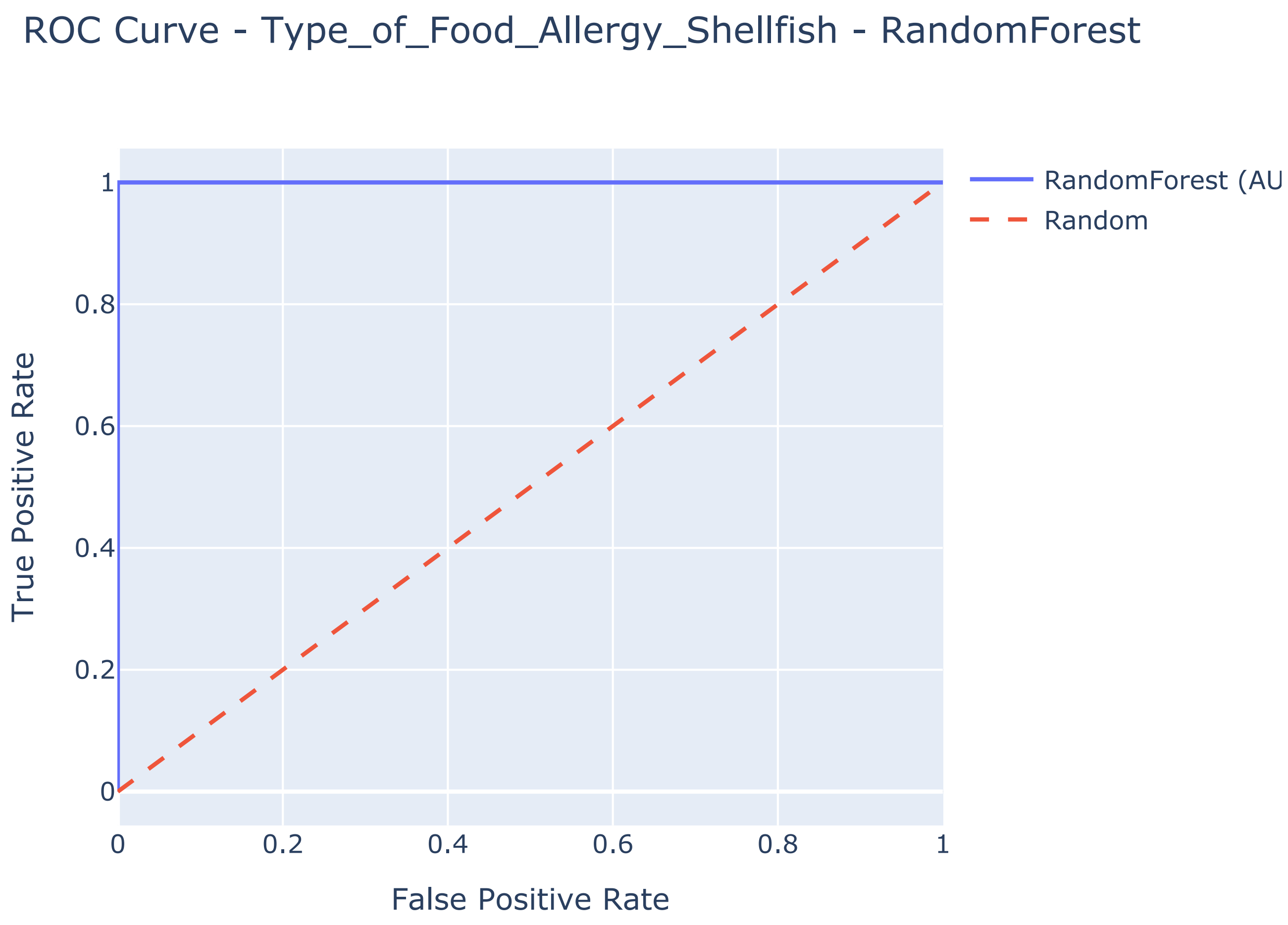
```
🔍 Target: Type_of_Food_Allergy_Peanut | Model: SVM
📈 Accuracy: 0.7369
🎯 F1 (0): 0.8377 | F1 (1): 0.2782
📊 Precision: 0.9020 | AUC: 0.7270146789247581
📊 Confusion Matrix:
 [[826   0]
 [ 68   0]]
```



ROC Curve - Type_of_Food_Allergy_Peanut - SVM

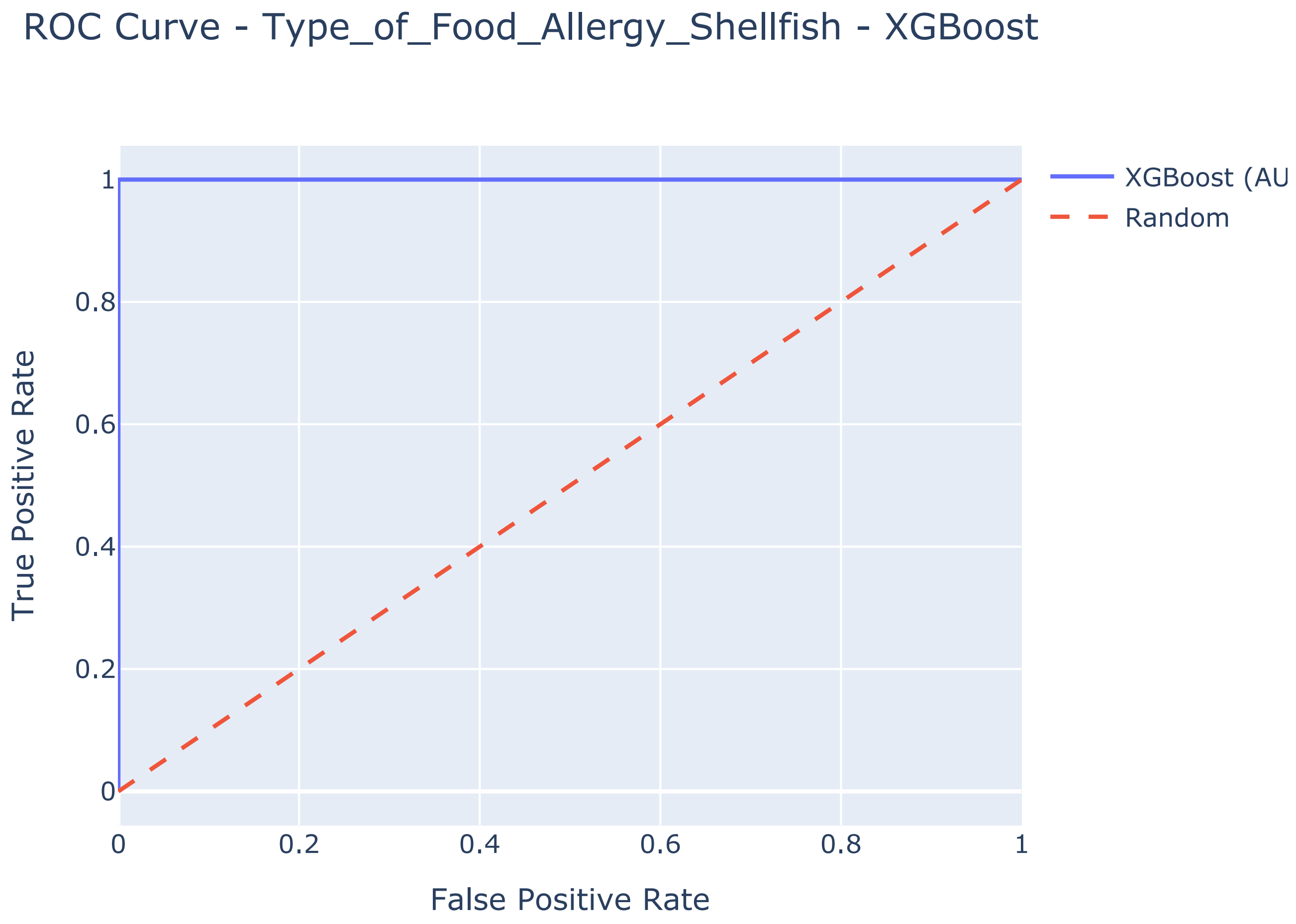


```
🔍 Target: Type_of_Food_Allergy_Shellfish | Model: RandomForest
📈 Accuracy: 0.9698
🎯 F1 (0): 0.9847 | F1 (1): 0.0500
📊 Precision: 0.9450 | AUC: 0.7830905283792213
📊 Confusion Matrix:
 [[867   0]
 [  0  27]]
```

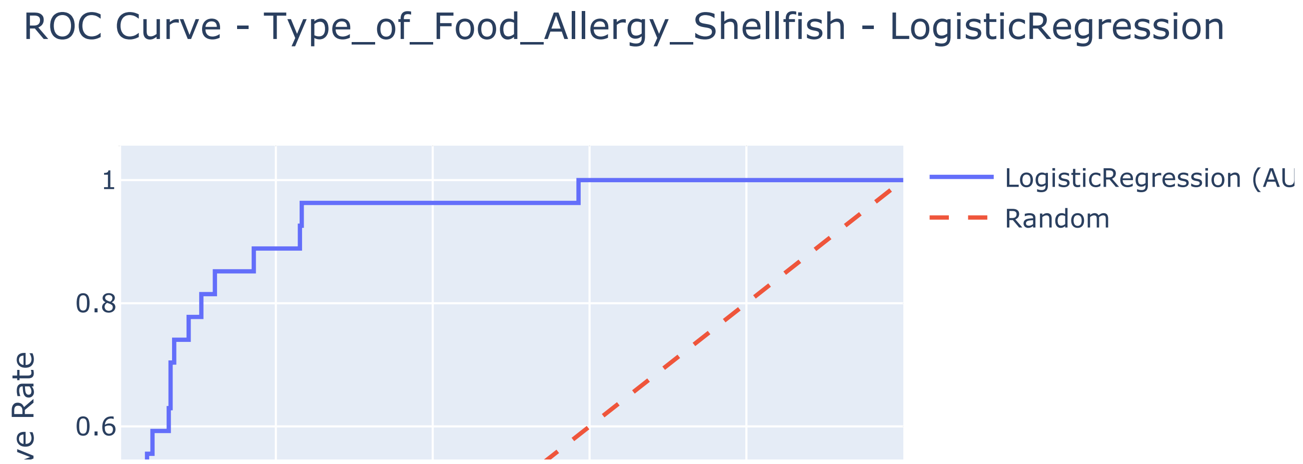


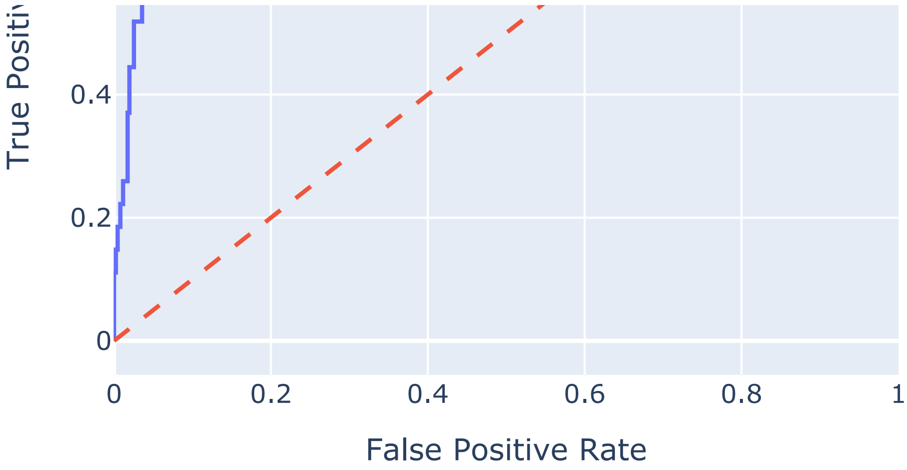
```
🔍 Target: Type_of_Food_Allergy_Shellfish | Model: XGBoost
📈 Accuracy: 0.9609
🎯 F1 (0): 0.9799 | F1 (1): 0.2000
📊 Precision: 0.9547 | AUC: 0.8007885592087678
📊 Confusion Matrix:
```

```
[[867   0]
 [  0  27]]
```

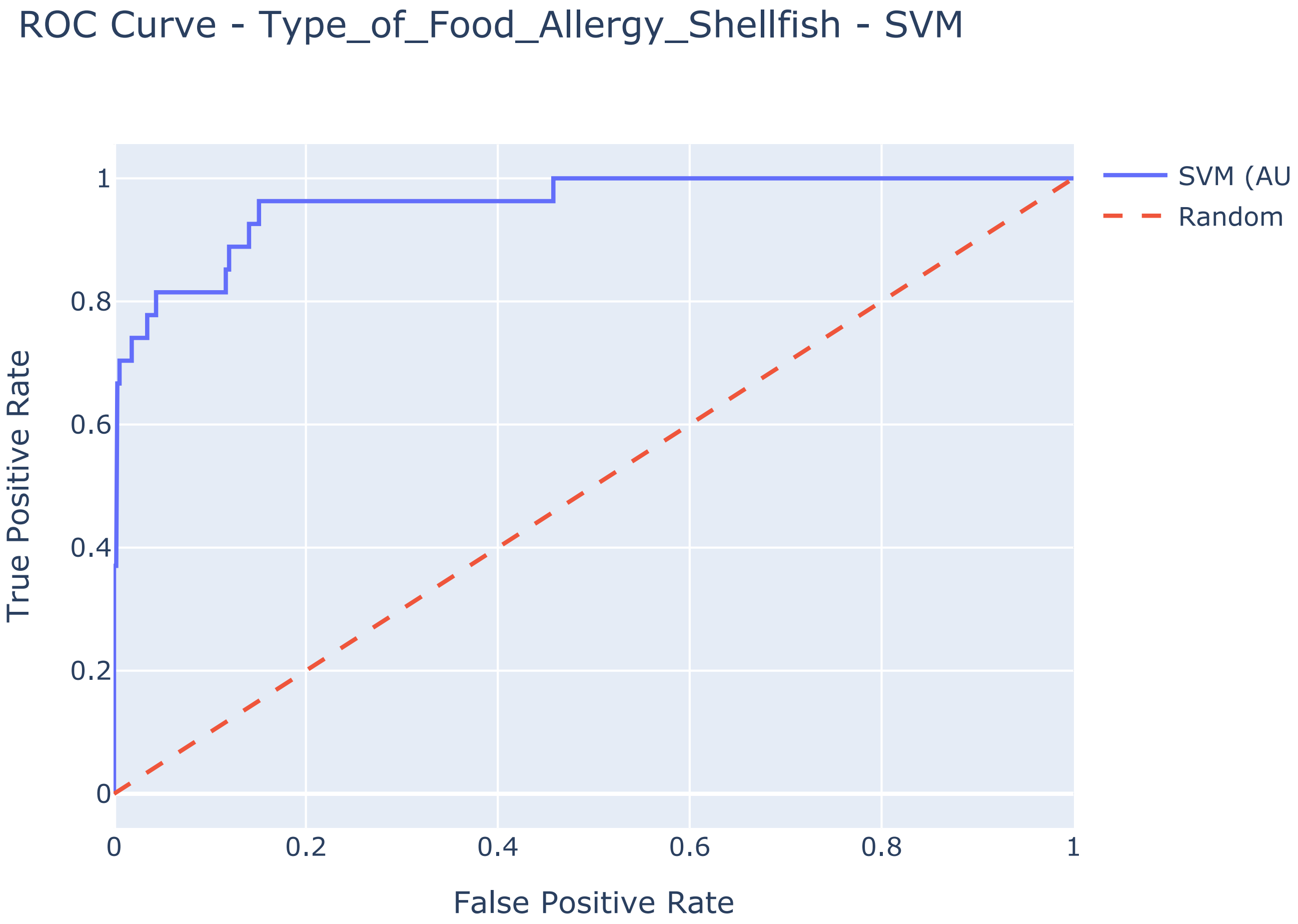


```
🔍 Target: Type_of_Food_Allergy_Shellfish | Model: LogisticRegression
📈 Accuracy: 0.9250
🎯 F1 (0): 0.9606 | F1 (1): 0.1430
📊 Precision: 0.9500 | AUC: 0.7110665597433841
📊 Confusion Matrix:
[[867   0]
 [ 27   0]]
```

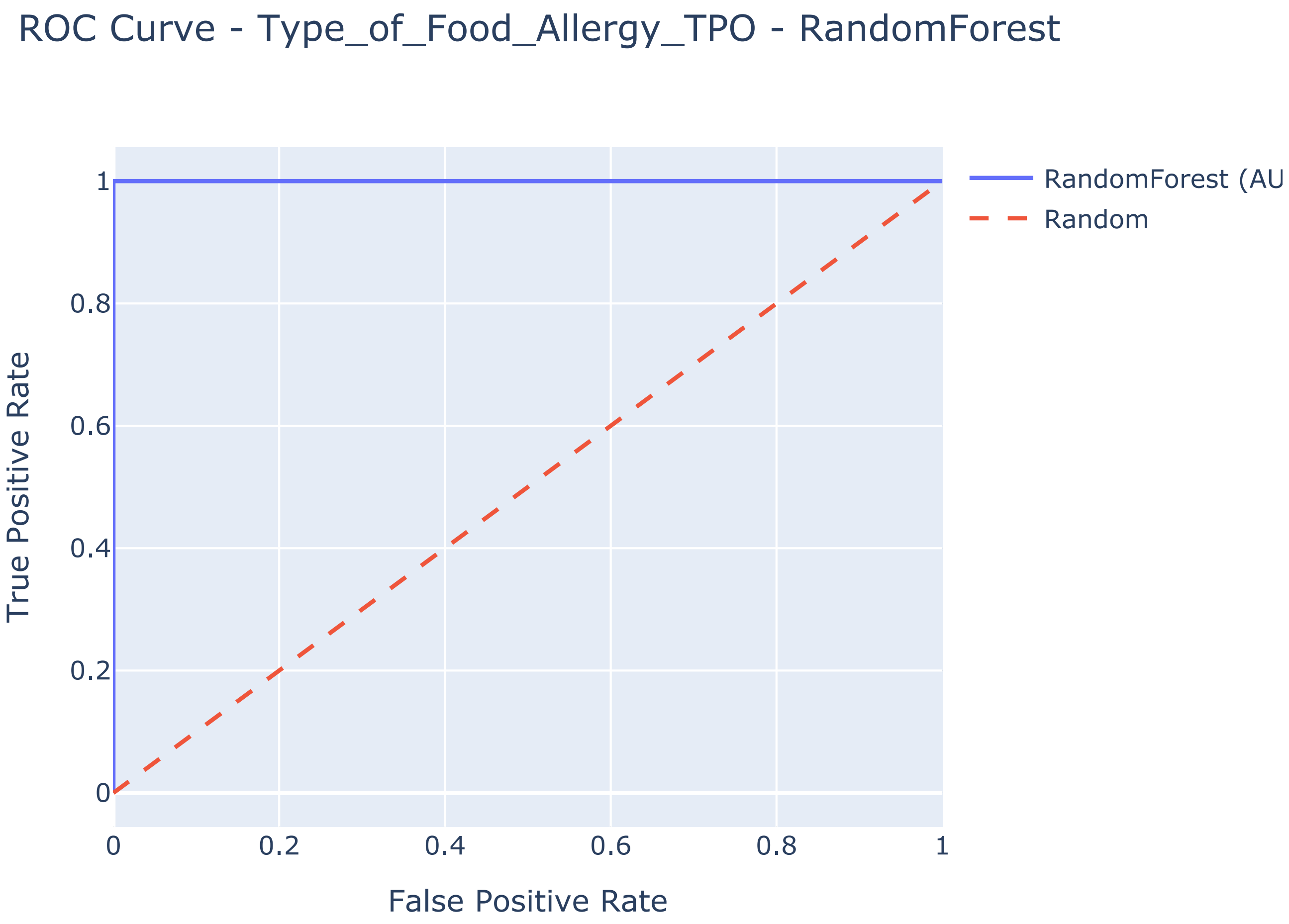




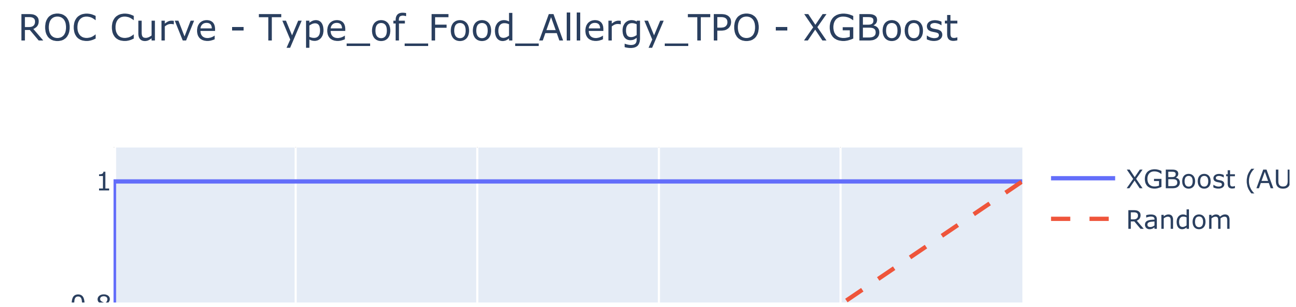
```
🔍 Target: Type_of_Food_Allergy_Shellfish | Model: SVM
📈 Accuracy: 0.7638
🎯 F1 (0): 0.8619 | F1 (1): 0.1352
📊 Precision: 0.9566 | AUC: 0.7325180433039294
📊 Confusion Matrix:
 [[867   0]
 [ 27   0]]
```



```
🔍 Target: Type_of_Food_Allergy_TPO | Model: RandomForest
🔍 Target: Type_of_Food_Allergy_TPO | Model: RandomForest
📈 Accuracy: 0.9396
🎯 F1 (0): 0.9685 | F1 (1): 0.2317
📊 Precision: 0.9247 | AUC: 0.8500644257703082
📊 Confusion Matrix:
 [[848   0]
 [  0  46]]
```

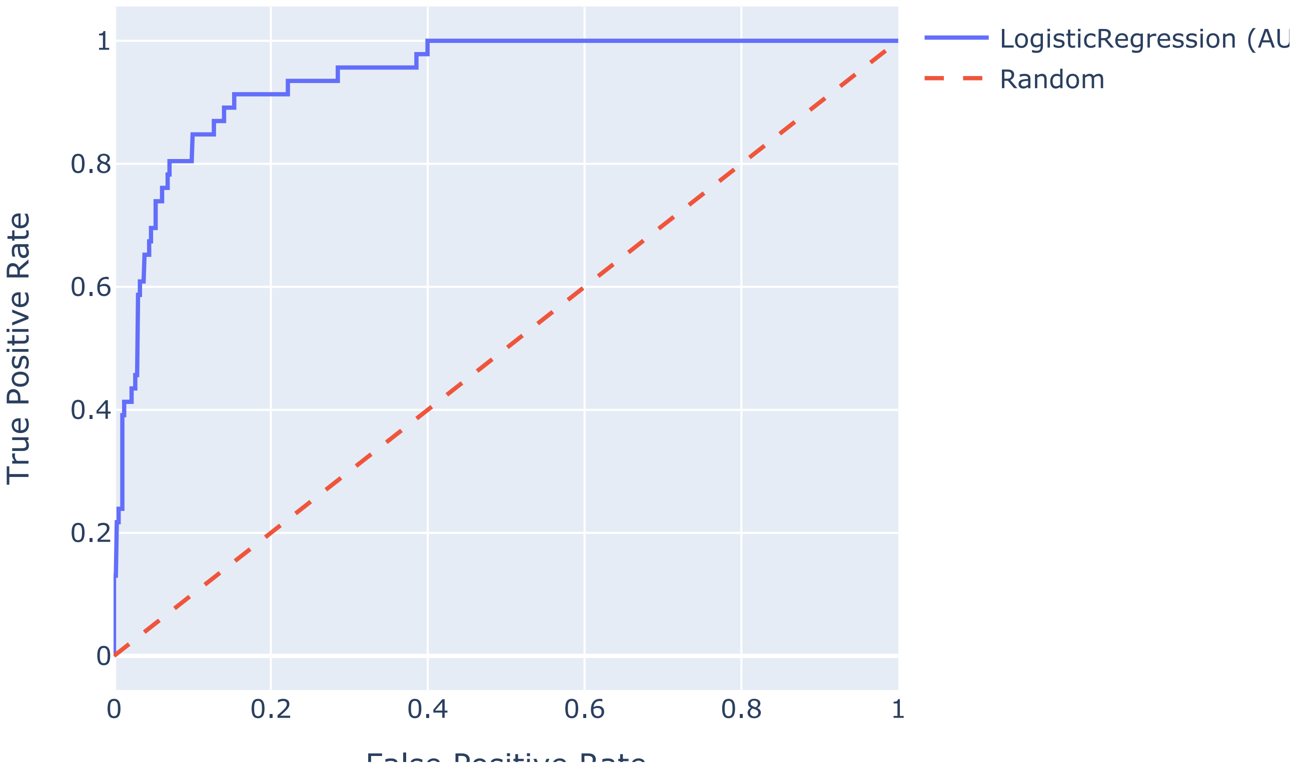


```
🔍 Target: Type_of_Food_Allergy_TPO | Model: XGBoost
📈 Accuracy: 0.9317
🎯 F1 (0): 0.9641 | F1 (1): 0.2683
📊 Precision: 0.9277 | AUC: 0.8226974789915967
📊 Confusion Matrix:
 [[848   0]
 [  0  46]]
```



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

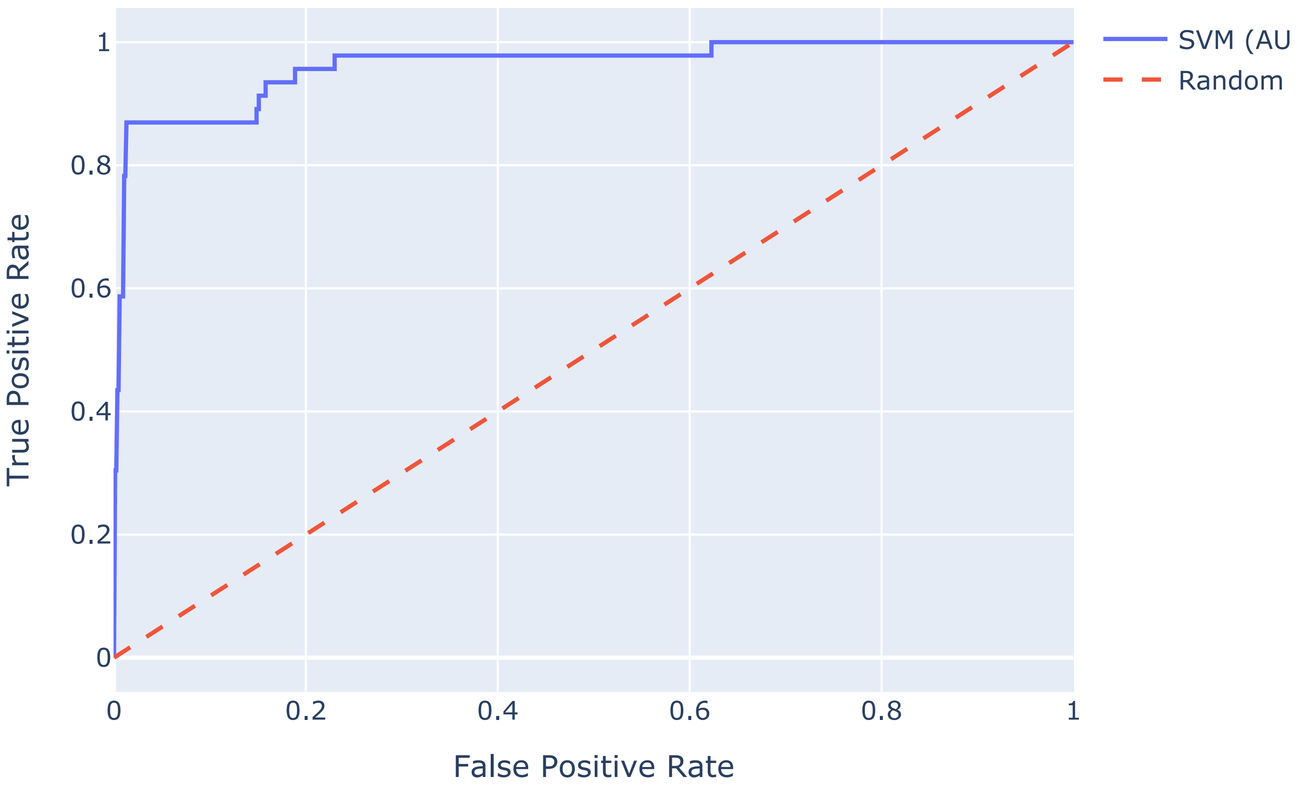
```
🔍 Target: Type_of_Food_Allergy_TPO | Model: LogisticRegression
📈 Accuracy: 0.9127
🎯 F1 (0): 0.9536 | F1 (1): 0.2230
📊 Precision: 0.9231 | AUC: 0.7879887955182073
📊 Confusion Matrix:
 [[845   3]
 [ 36  10]]
```


ROC Curve - Type_of_Food_Allergy_TPO - LogisticRegression

False Positive Rate

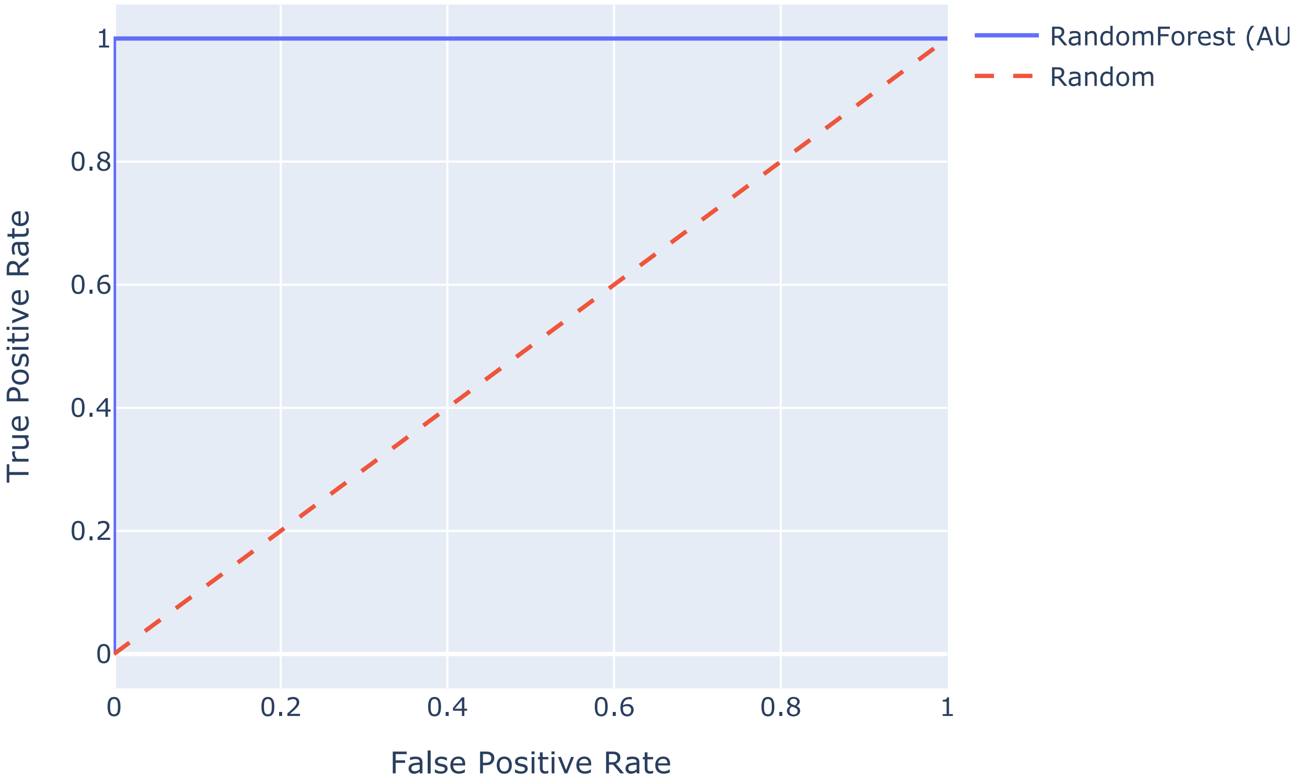
```
🔍 Target: Type_of_Food_Allergy_TPO | Model: SVM
📈 Accuracy: 0.8064
🎯 F1 (0): 0.8873 | F1 (1): 0.2910
📊 Precision: 0.9433 | AUC: 0.8249019607843138
📊 Confusion Matrix:
 [[848   0]
 [ 46   0]]
```

ROC Curve - Type_of_Food_Allergy_TPO - SVM



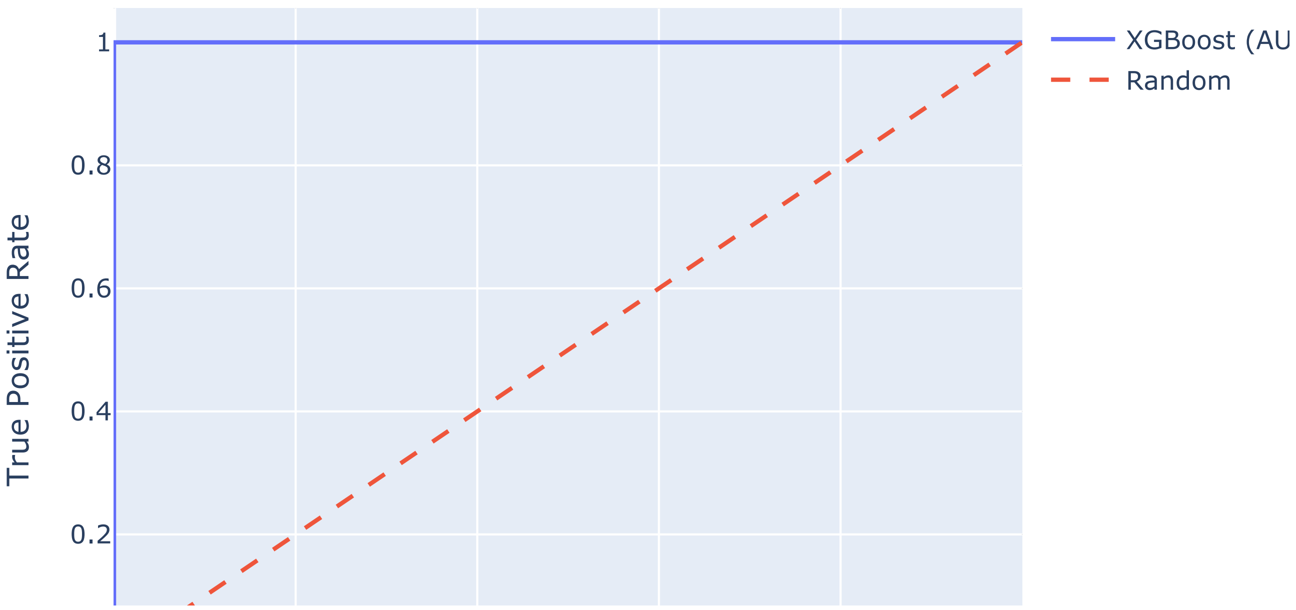
```
🔍 Target: Type_of_Food_Allergy_Tree_Nuts | Model: RandomForest
📈 Accuracy: 0.9172
🎯 F1 (0): 0.9560 | F1 (1): 0.2820
📊 Precision: 0.8992 | AUC: 0.8458188153310104
📊 Confusion Matrix:
 [[820   0]
 [  0  74]]
```

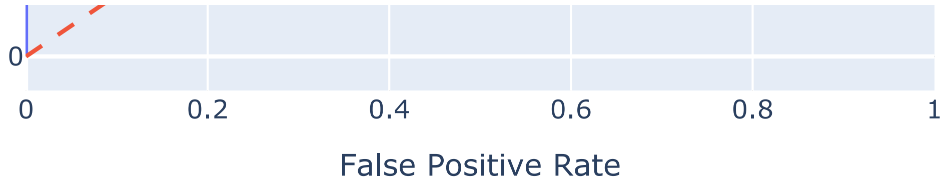
ROC Curve - Type_of_Food_Allergy_Tree_Nuts - RandomForest



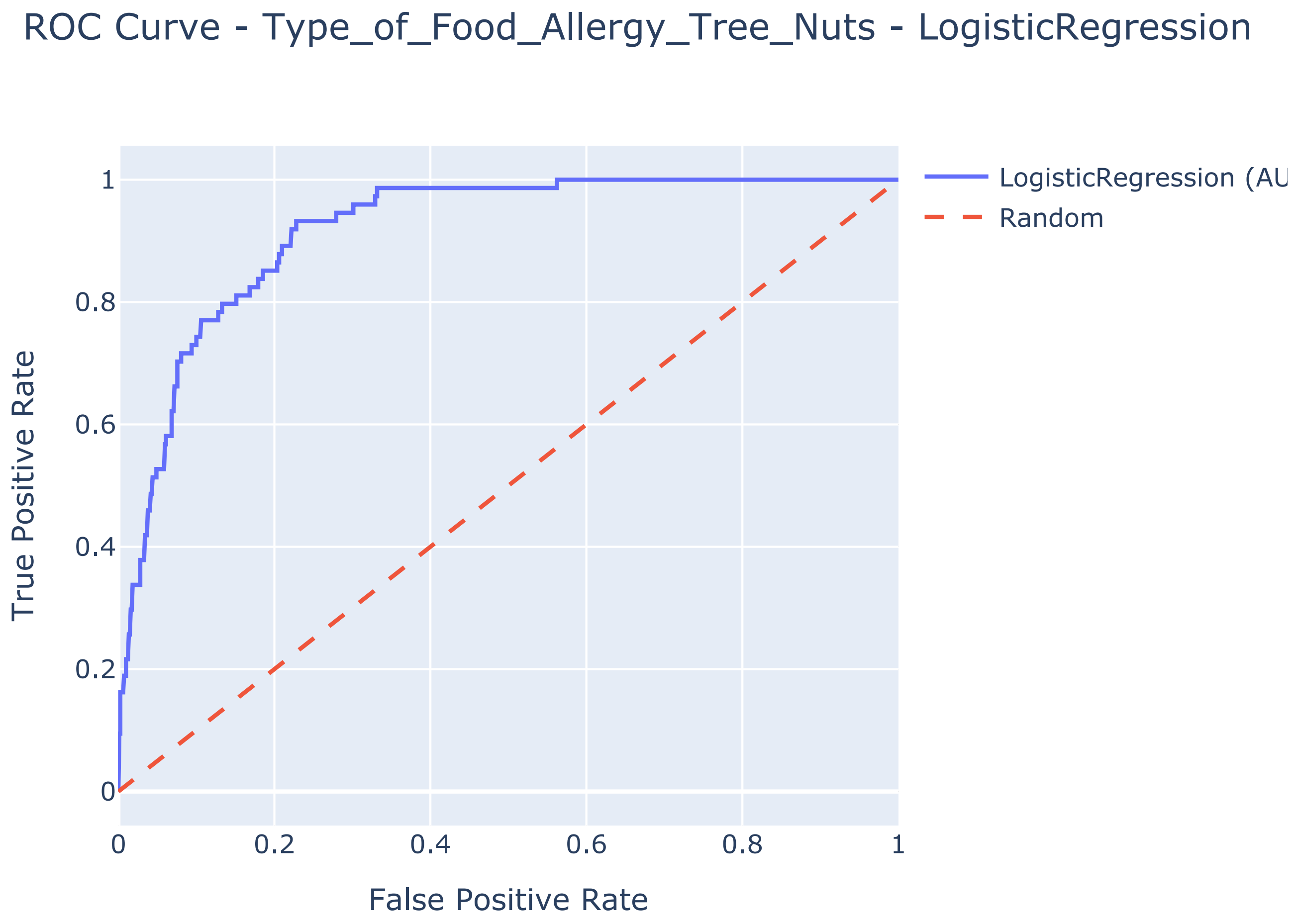
```
🔍 Target: Type_of_Food_Allergy_Tree_Nuts | Model: XGBoost
📈 Accuracy: 0.9183
🎯 F1 (0): 0.9560 | F1 (1): 0.4085
📊 Precision: 0.9070 | AUC: 0.8562282229965158
📊 Confusion Matrix:
 [[820   0]
 [  0  74]]
```

ROC Curve - Type_of_Food_Allergy_Tree_Nuts - XGBoost



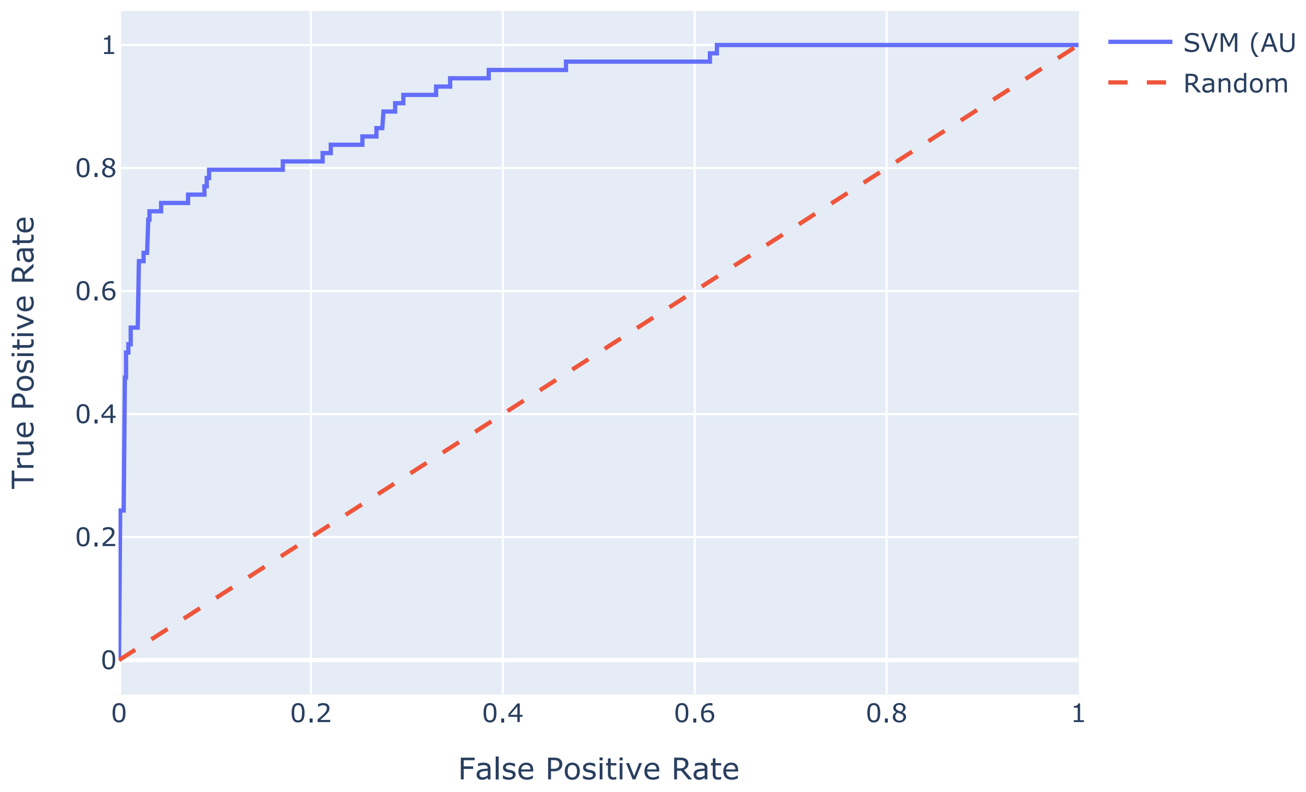


```
🔍 Target: Type_of_Food_Allergy_Tree_Nuts | Model: LogisticRegression
📈 Accuracy: 0.8681
🎯 F1 (0): 0.9271 | F1 (1): 0.2944
📊 Precision: 0.8830 | AUC: 0.7658101045296167
📊 Confusion Matrix:
 [[813   7]
 [ 60  14]]
```



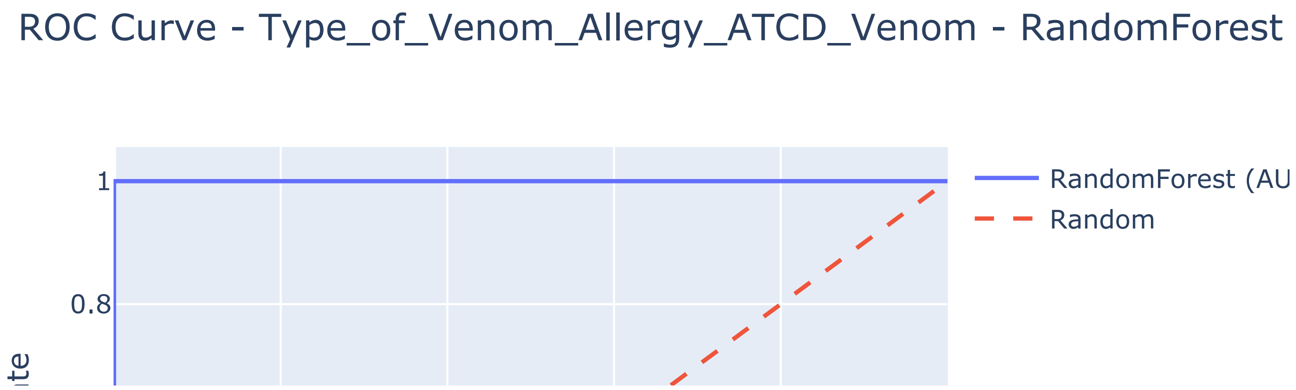
```
🔍 Target: Type_of_Food_Allergy_Tree_Nuts | Model: SVM
📈 Accuracy: 0.7002
🎯 F1 (0): 0.8106 | F1 (1): 0.2641
📊 Precision: 0.8922 | AUC: 0.7194033101045296
📊 Confusion Matrix:
 [[820   0]
 [ 74   0]]
```

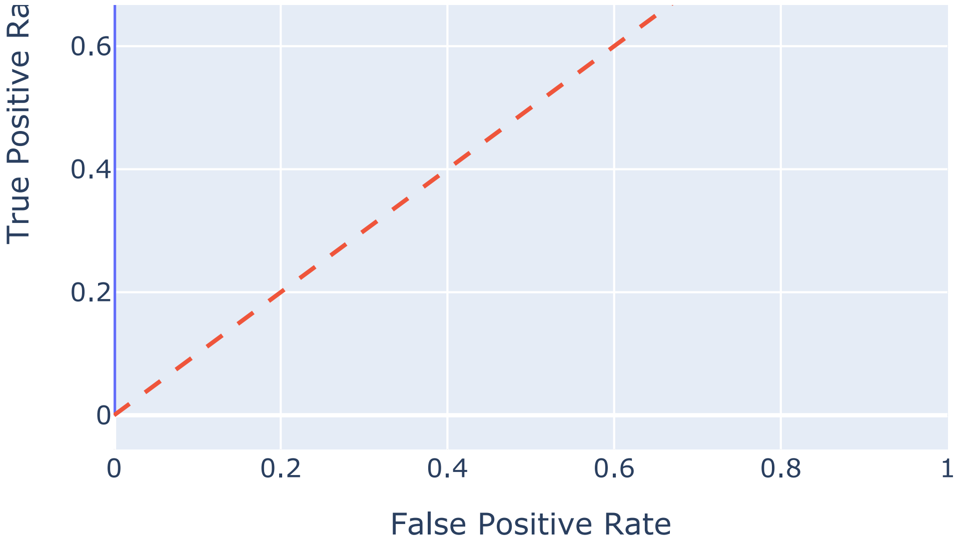
ROC Curve - Type_of_Food_Allergy_Tree_Nuts - SVM



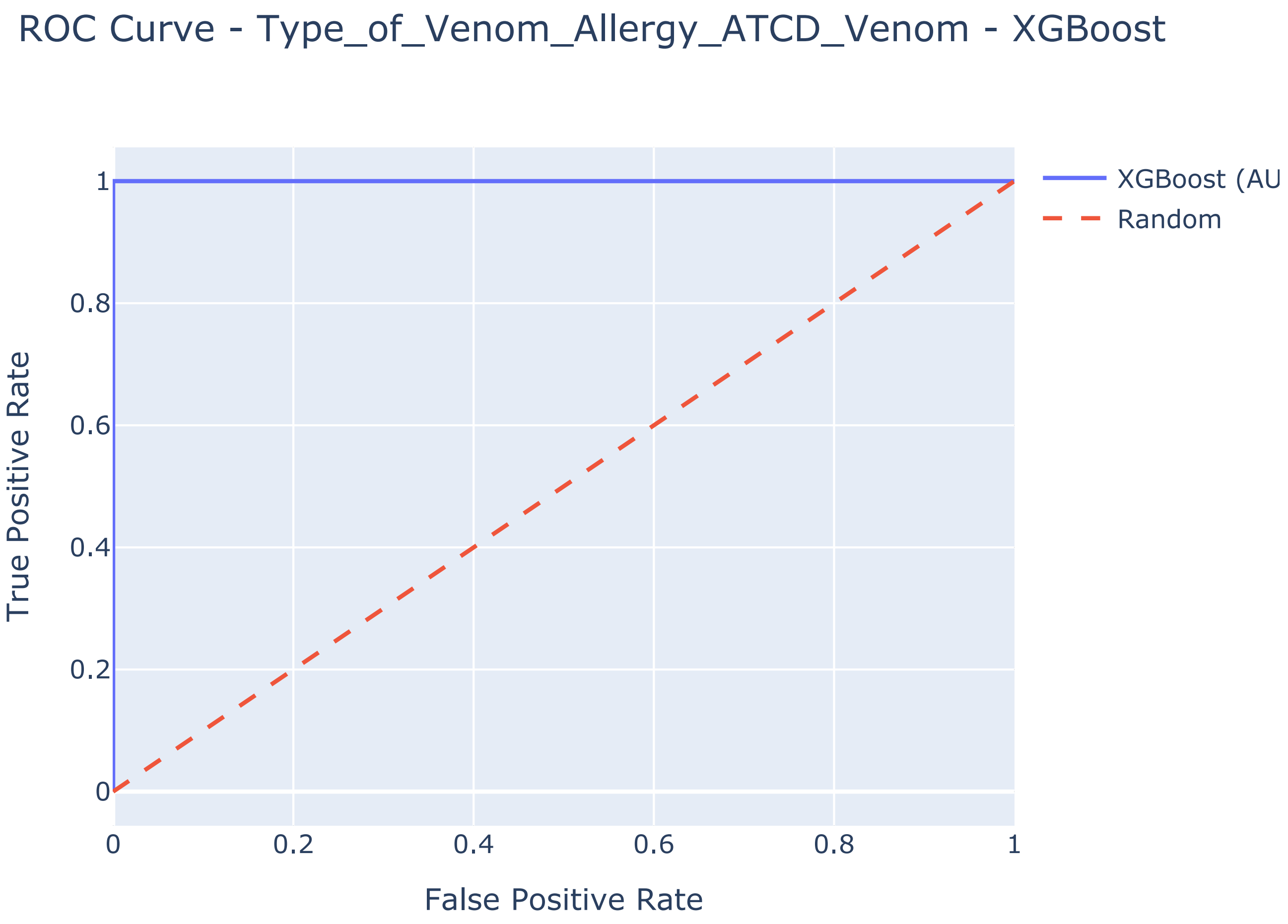
```
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedKFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.metrics import (
    f1_score, 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

V1_venom = V1[V1["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=V1_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 = V1_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_V1_venom.csv", index=False)
```

```
🔍 Target: Type_of_Venom_Allergy_ATCD_Venom | Model: RandomForest
📈 Accuracy: 0.8944
🎯 F1 (0): 0.9354 | F1 (1): 0.6500
📊 Precision: 0.9175 | AUC: 0.9125
📊 Confusion Matrix:
 [[81  0]
 [ 0 16]]
```

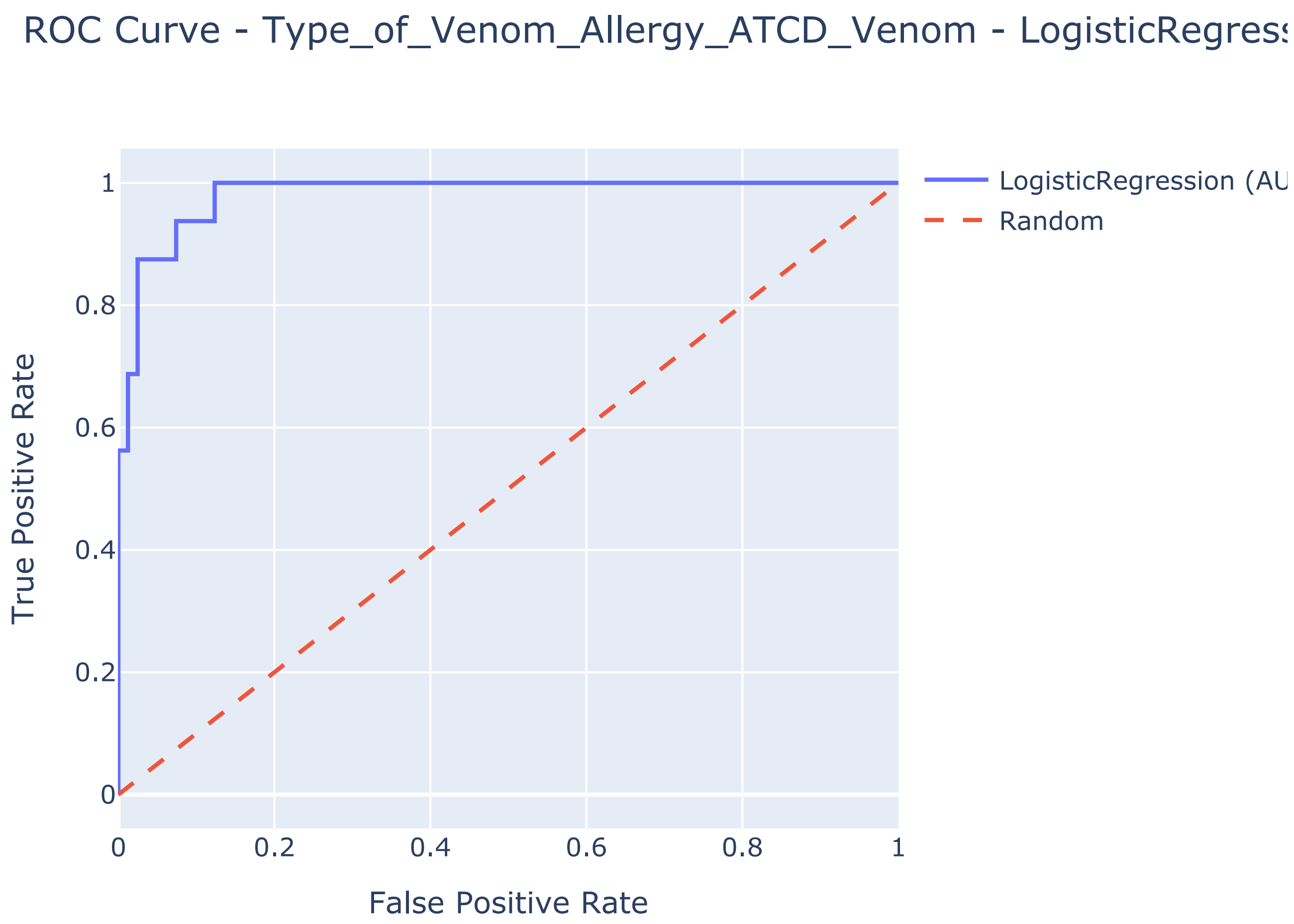




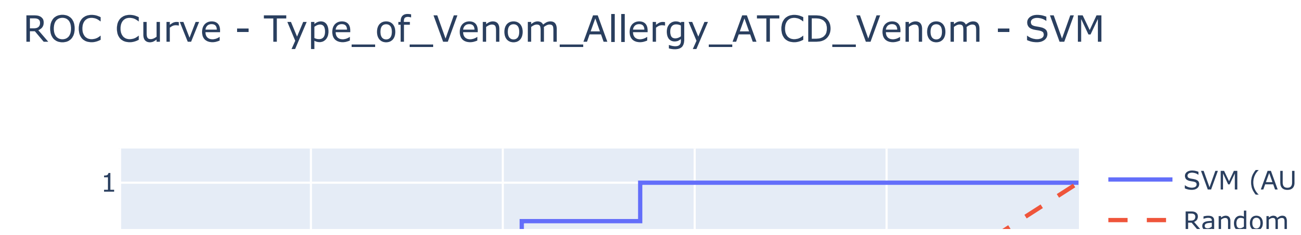
```
🔍 Target: Type_of_Venom_Allergy_ATCD_Venom | Model: XGBoost
📈 Accuracy: 0.8744
🎯 F1 (0): 0.9212 | F1 (1): 0.6200
📊 Precision: 0.9061 | AUC: 0.90625
📊 Confusion Matrix:
 [[81  0]
 [ 0 16]]
```

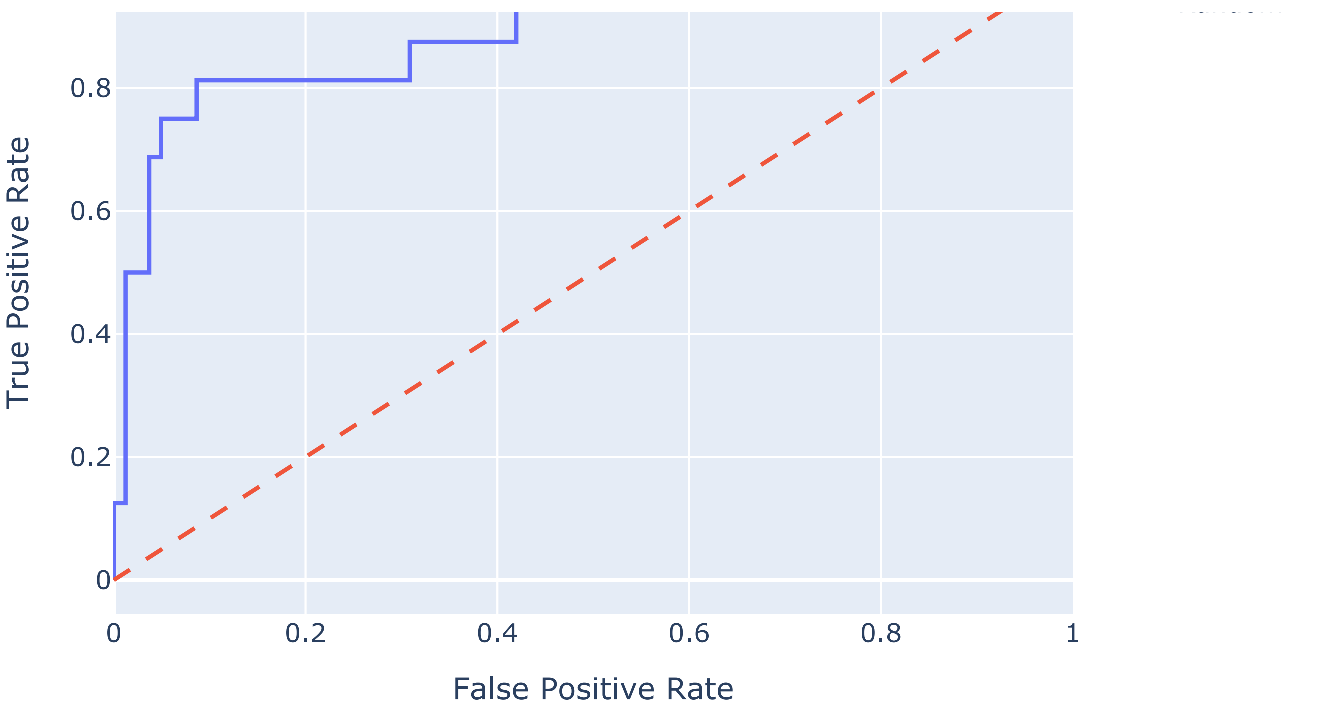


```
🔍 Target: Type_of_Venom_Allergy_ATCD_Venom | Model: LogisticRegression
📈 Accuracy: 0.8244
🎯 F1 (0): 0.8919 | F1 (1): 0.4867
📊 Precision: 0.8507 | AUC: 0.8638888888888889
📊 Confusion Matrix:
 [[79  2]
 [ 3 13]]
```

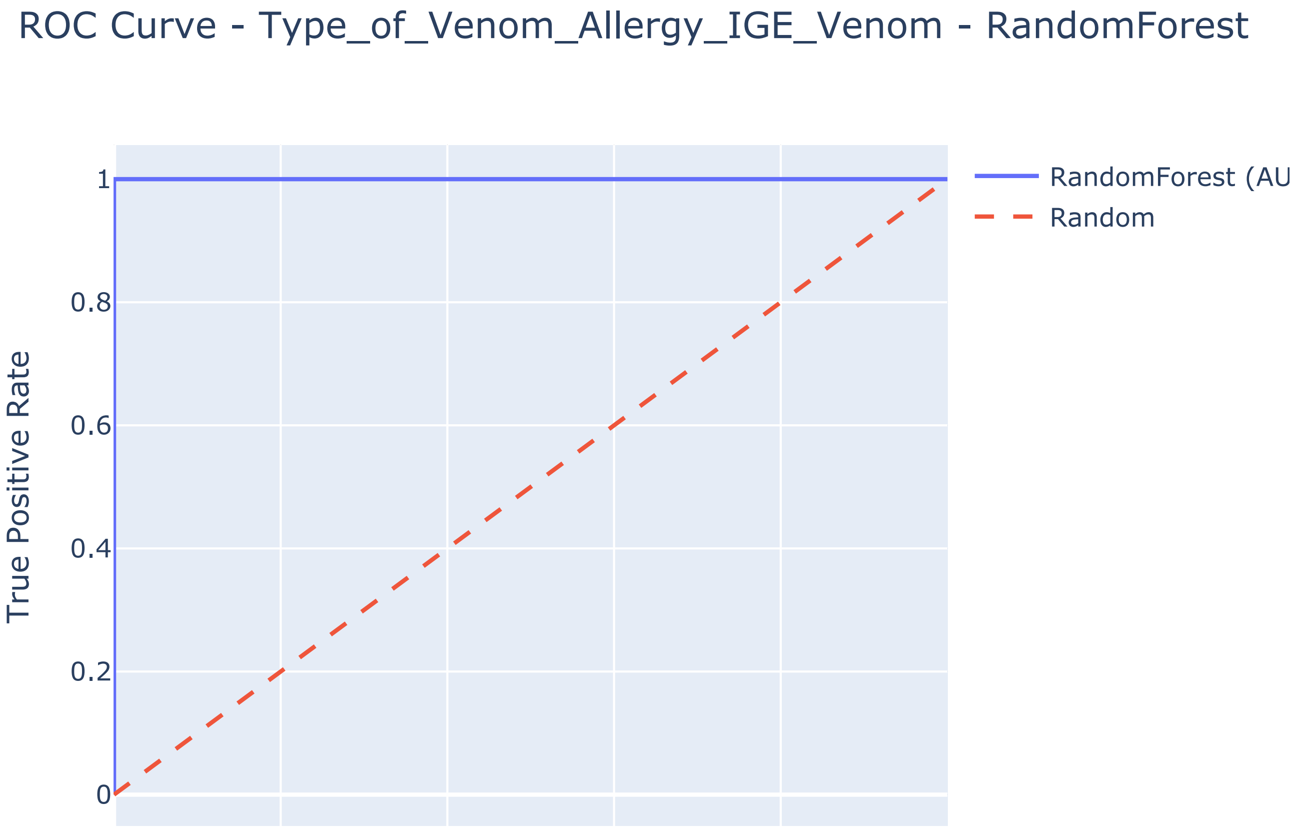


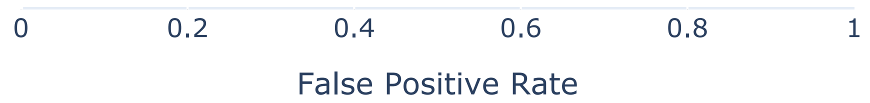
```
🔍 Target: Type_of_Venom_Allergy_ATCD_Venom | Model: SVM
📈 Accuracy: 0.6822
🎯 F1 (0): 0.7608 | F1 (1): 0.4655
📊 Precision: 0.8720 | AUC: 0.9027777777777779
📊 Confusion Matrix:
 [[81  0]
 [16  0]]
```





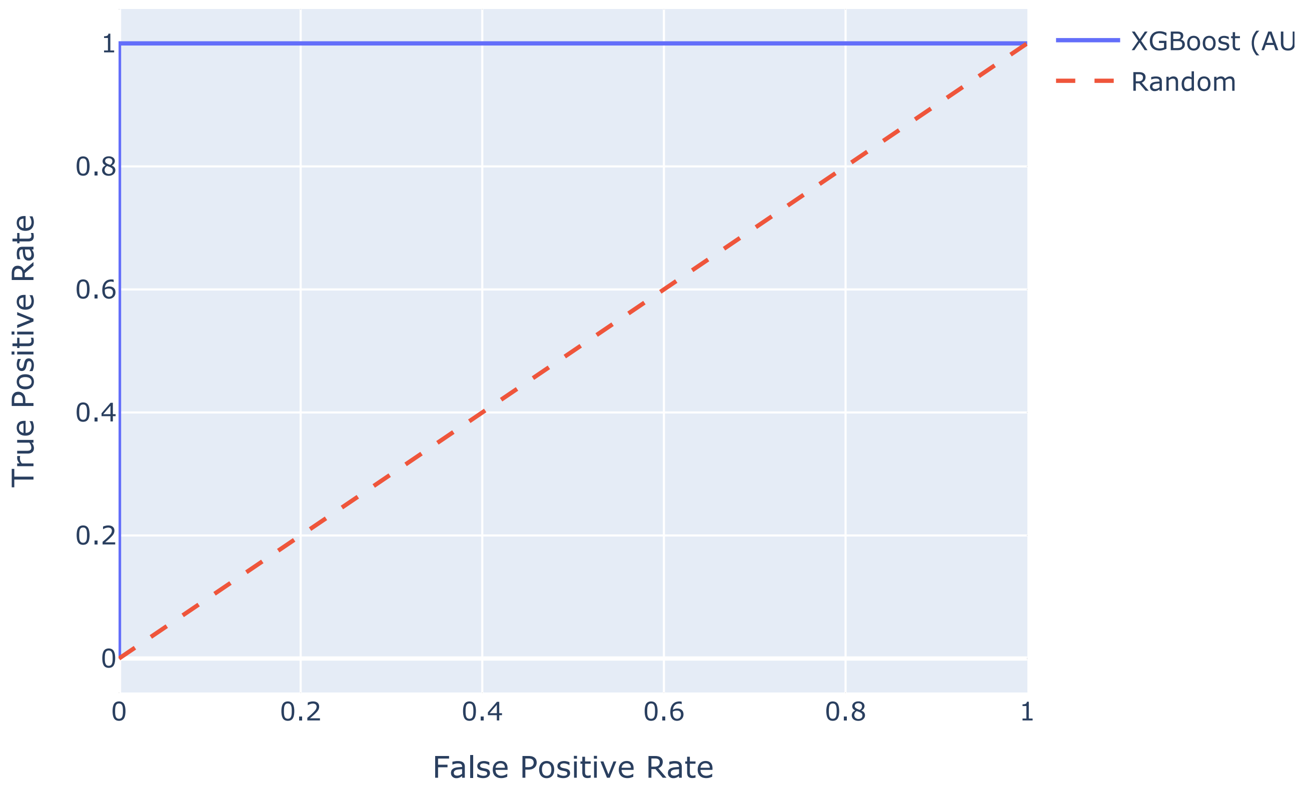
```
🔍 Target: Type_of_Venom_Allergy_IGE_Venom | Model: RandomForest
📈 Accuracy: 0.9900
🎯 F1 (0): 0.9000 | F1 (1): 0.9947
📊 Precision: 0.9810 | AUC: 1.0
📊 Confusion Matrix:
 [[10  0]
 [ 0 87]]
```





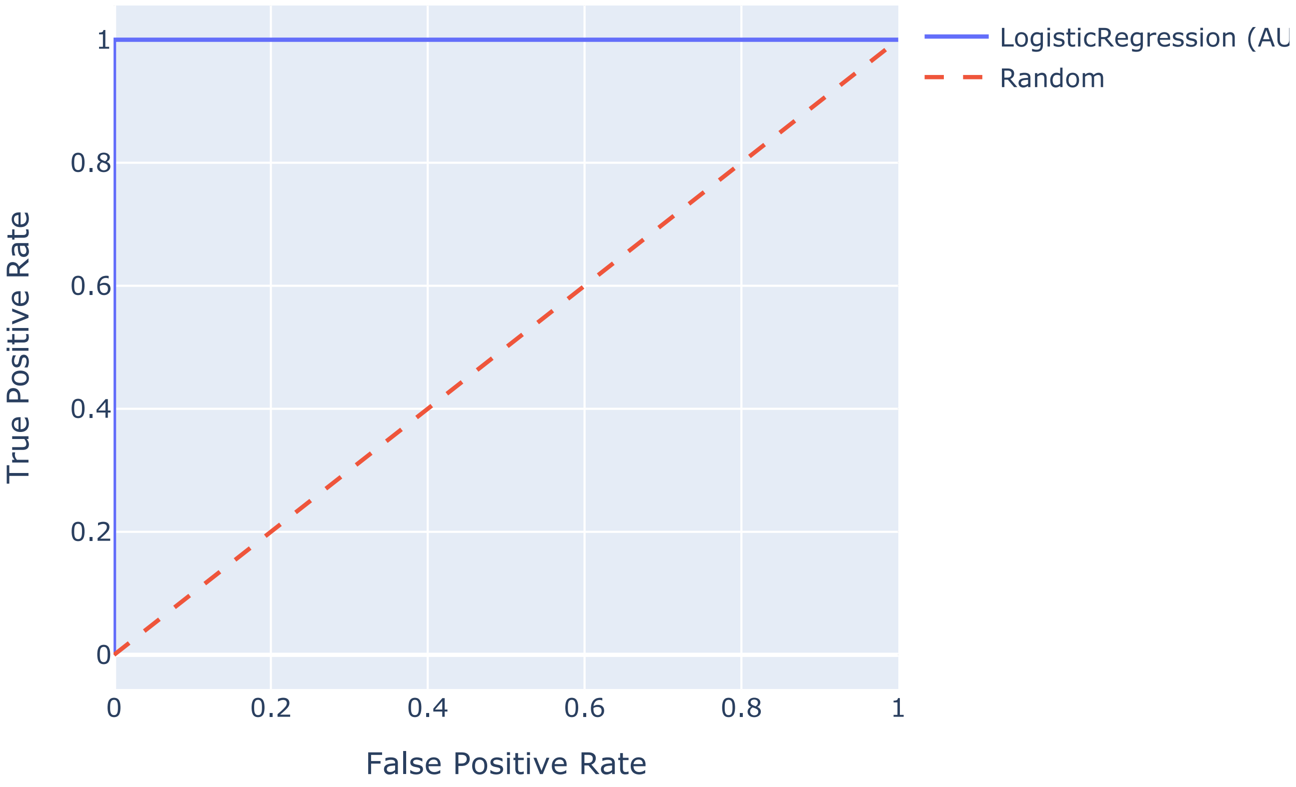
```
🔍 Target: Type_of_Venom_Allergy_IGE_Venom | Model: XGBoost
📈 Accuracy: 1.0000
🎯 F1 (0): 1.0000 | F1 (1): 1.0000
📊 Precision: 1.0000 | AUC: 1.0
📊 Confusion Matrix:
 [[10  0]
 [ 0 87]]
```

ROC Curve - Type_of_Venom_Allergy_IGE_Venom - XGBoost



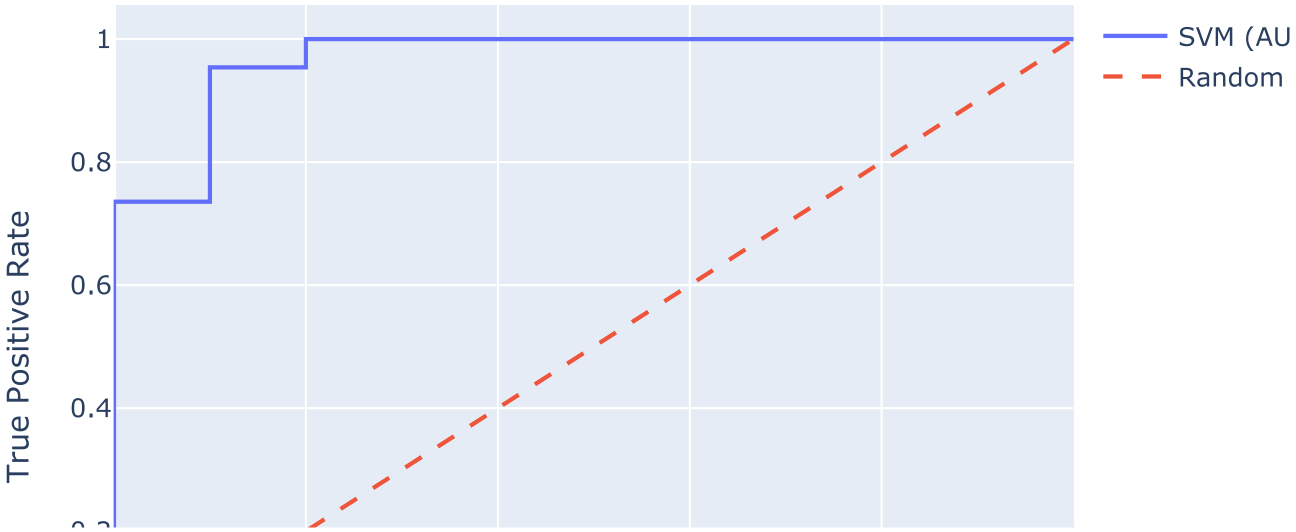
```
🔍 Target: Type_of_Venom_Allergy_IGE_Venom | Model: LogisticRegression
📈 Accuracy: 0.8622
🎯 F1 (0): 0.3833 | F1 (1): 0.9198
📊 Precision: 0.8786 | AUC: 0.9041666666666666
📊 Confusion Matrix:
 [[10  0]
 [ 0 87]]
```

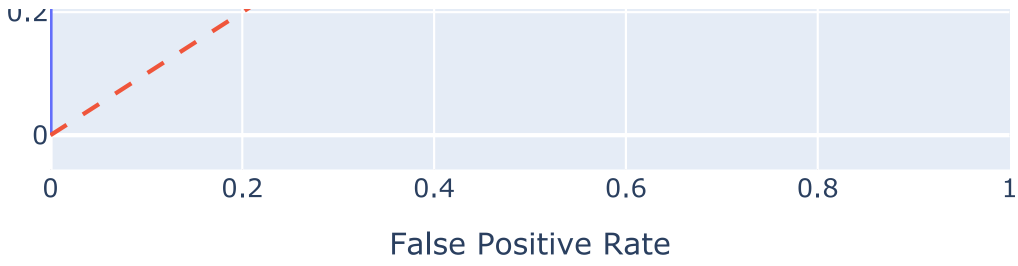
ROC Curve - Type_of_Venom_Allergy_IGE_Venom - LogisticRegressic



```
🔍 Target: Type_of_Venom_Allergy_IGE_Venom | Model: SVM
📈 Accuracy: 0.6756
🎯 F1 (0): 0.3086 | F1 (1): 0.7755
📊 Precision: 0.8740 | AUC: 0.8458333333333332
📊 Confusion Matrix:
 [[ 0 10]
 [ 0 87]]
```

ROC Curve - Type_of_Venom_Allergy_IGE_Venom - SVM





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

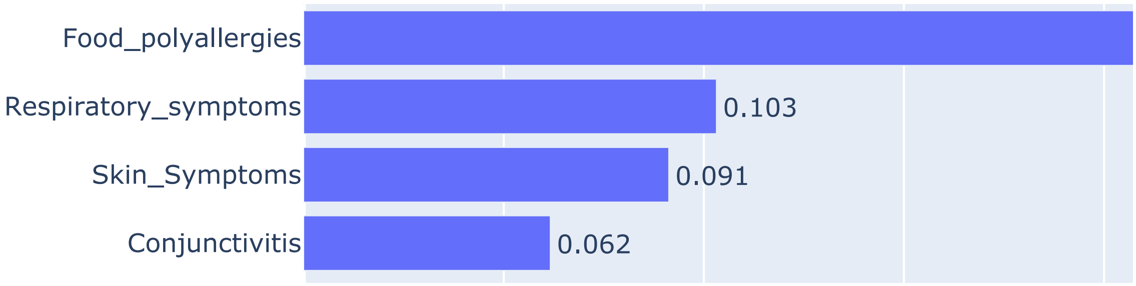
[ ] ↳ 1 cell hidden

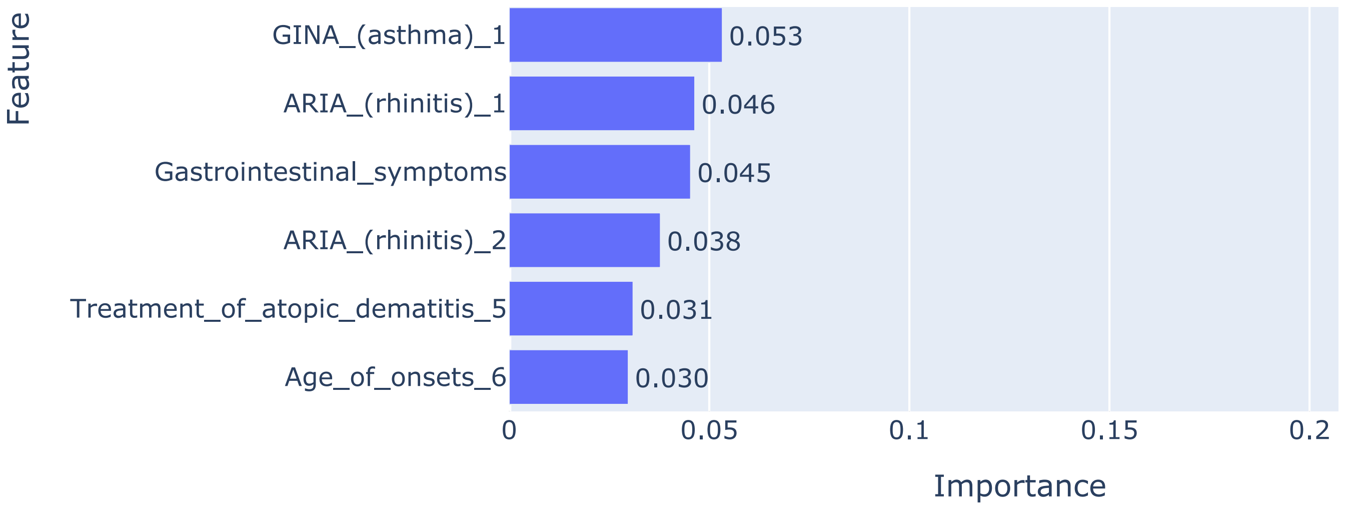
TOP Features

```
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 = V1.copy()
X.drop(target_1, axis=1, inplace=True)
X.drop(extra_columns, axis=1, inplace=True)
```

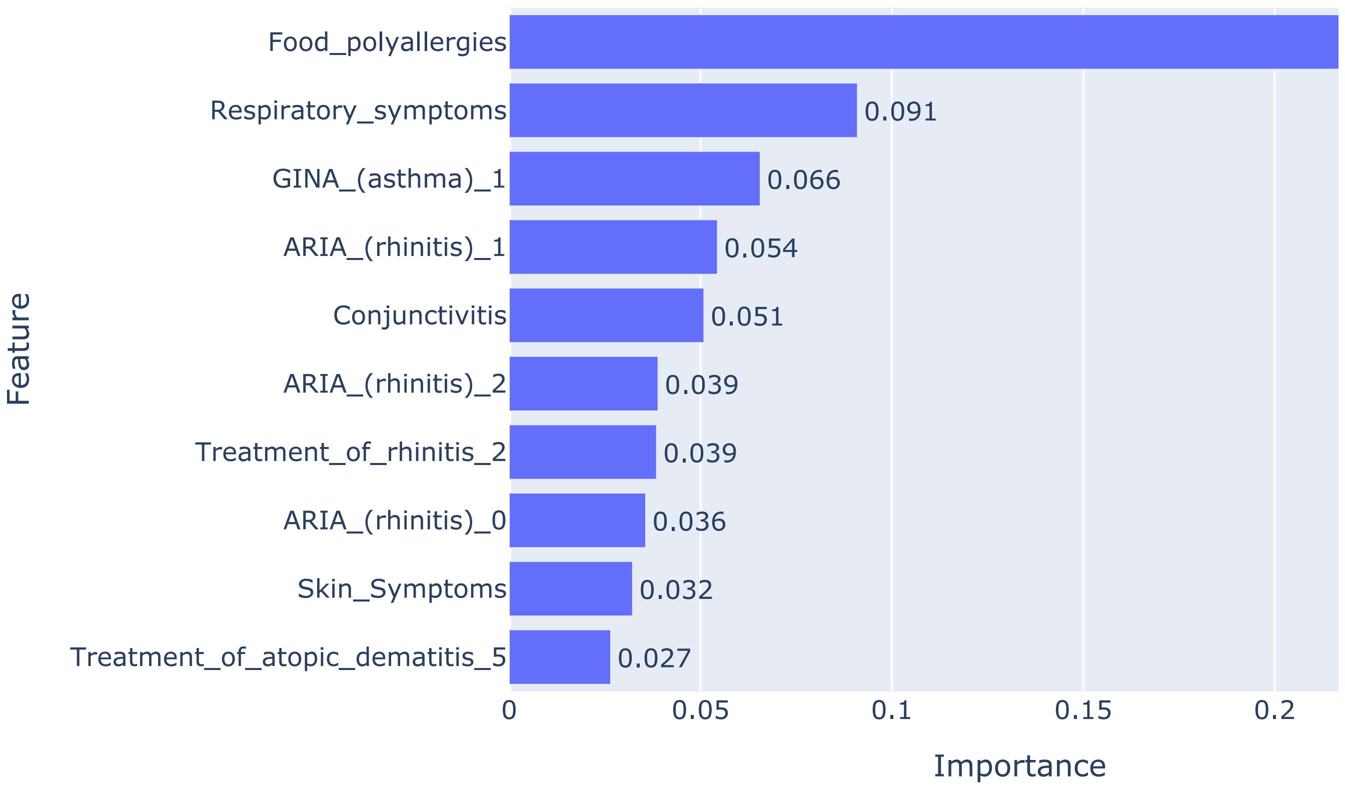
```
X.drop(extra, axis=1, inplace=True)
X.drop(inconnu, axis=1, inplace=True)
X = X.iloc[:, 1:]

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],
        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 = V1[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)



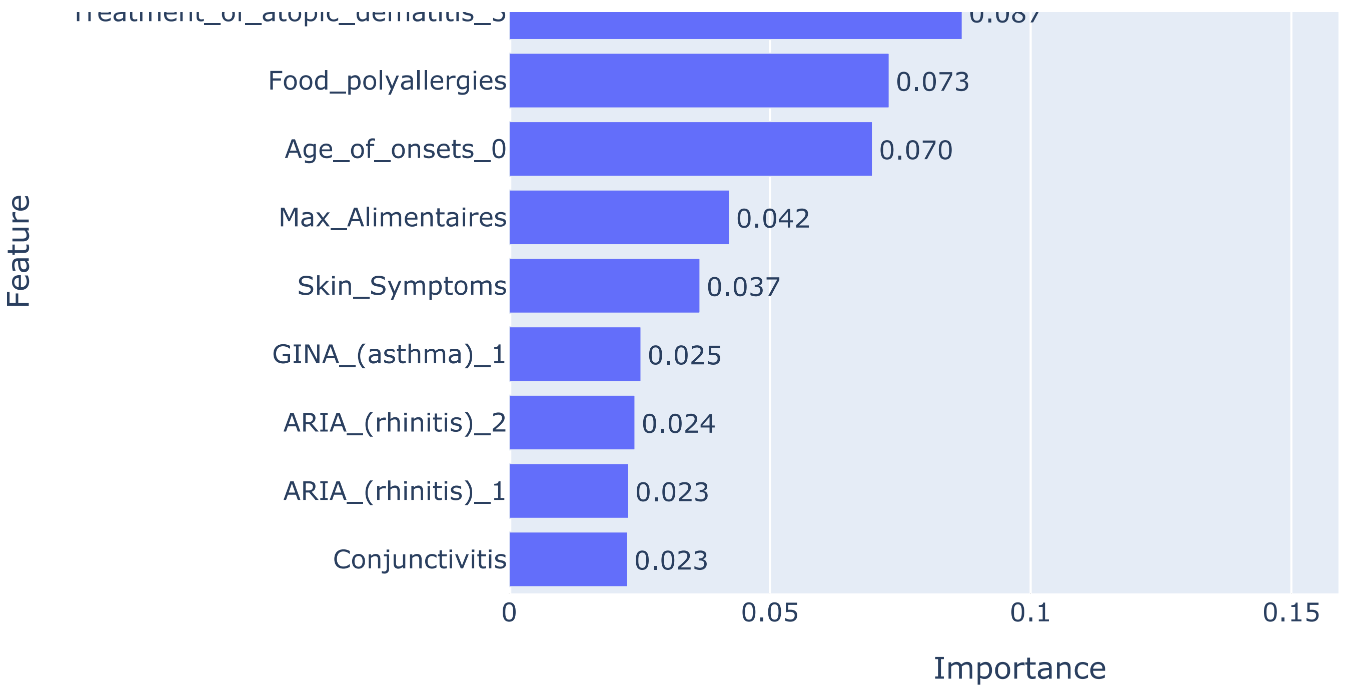


Top 10 Features pour la cible 'Respiratory_Allergy' (XGBoost)

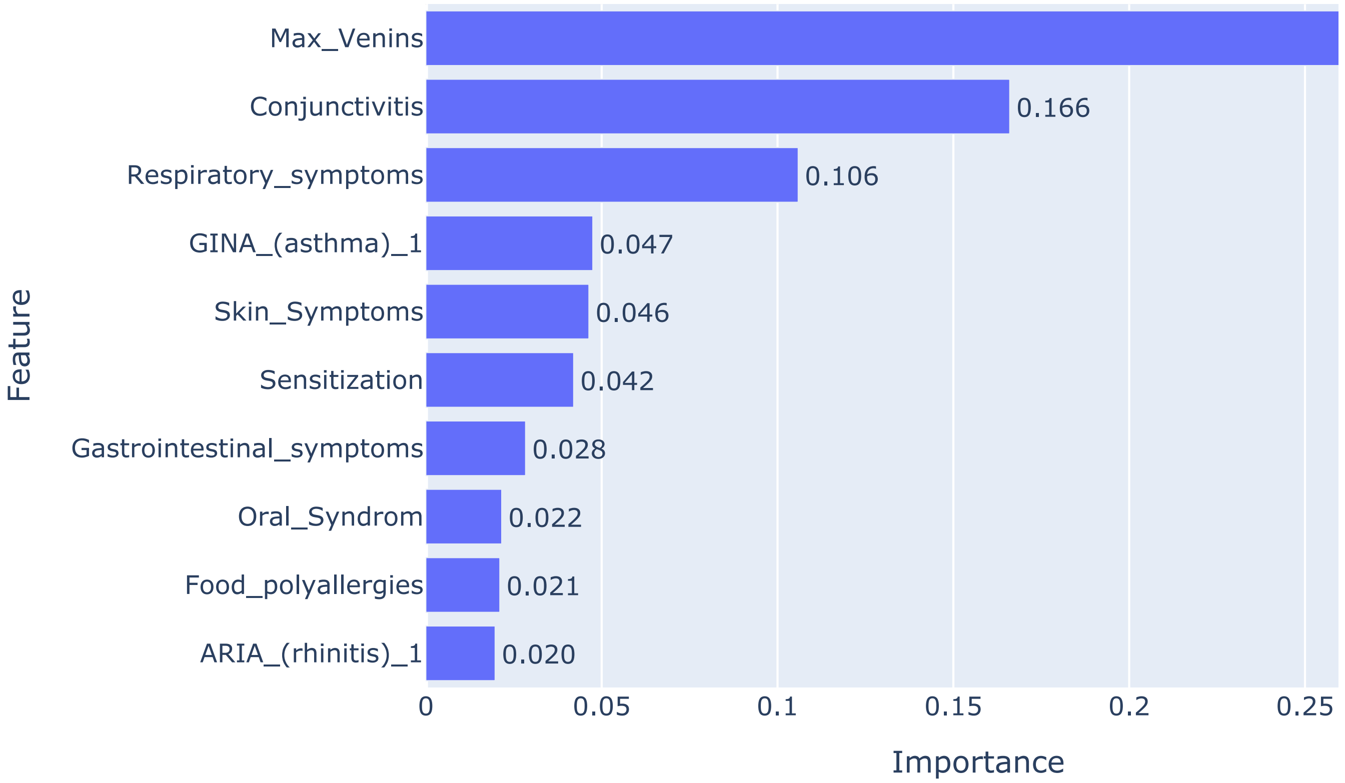


Top 10 Features pour la cible 'Food_Allergy' (XGBoost)

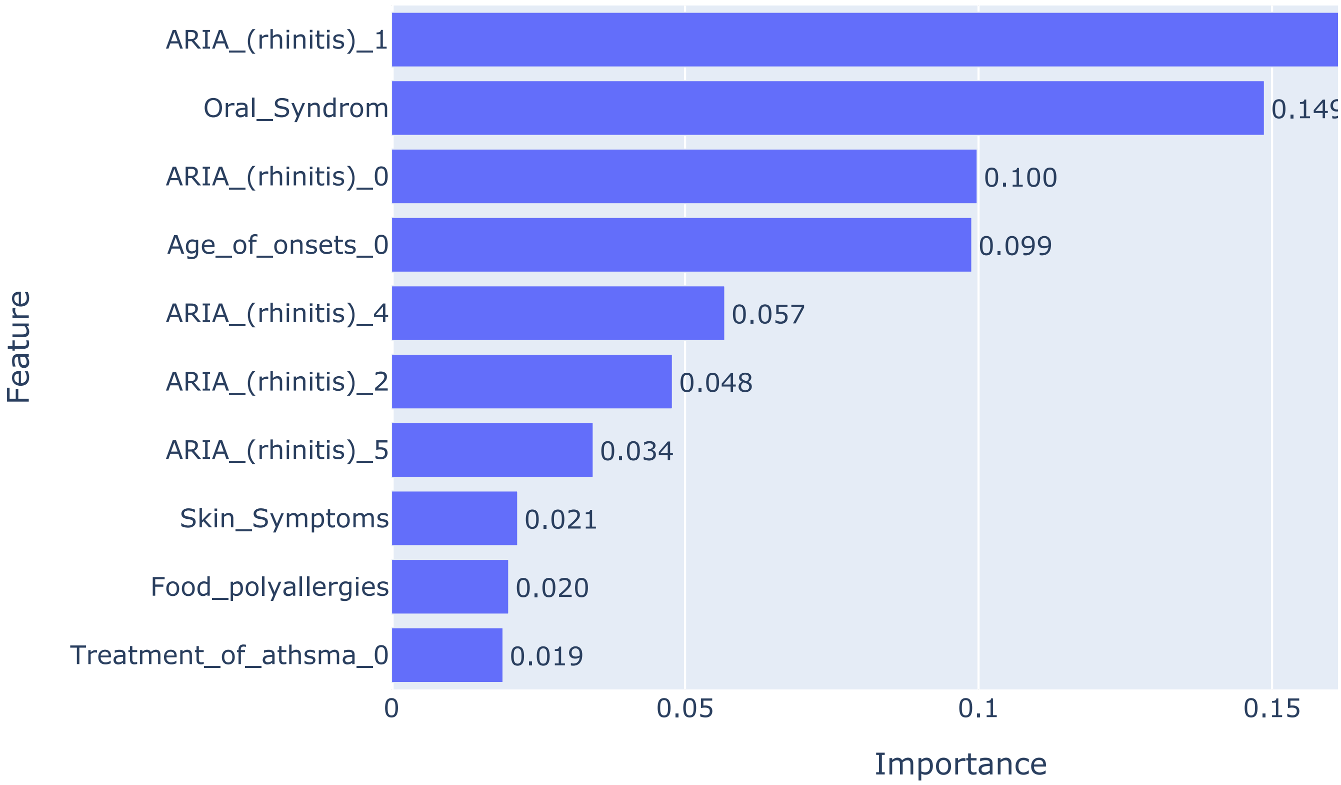
Respiratory_symptoms
Treatment_of_atopic_dematitis_5 0.087



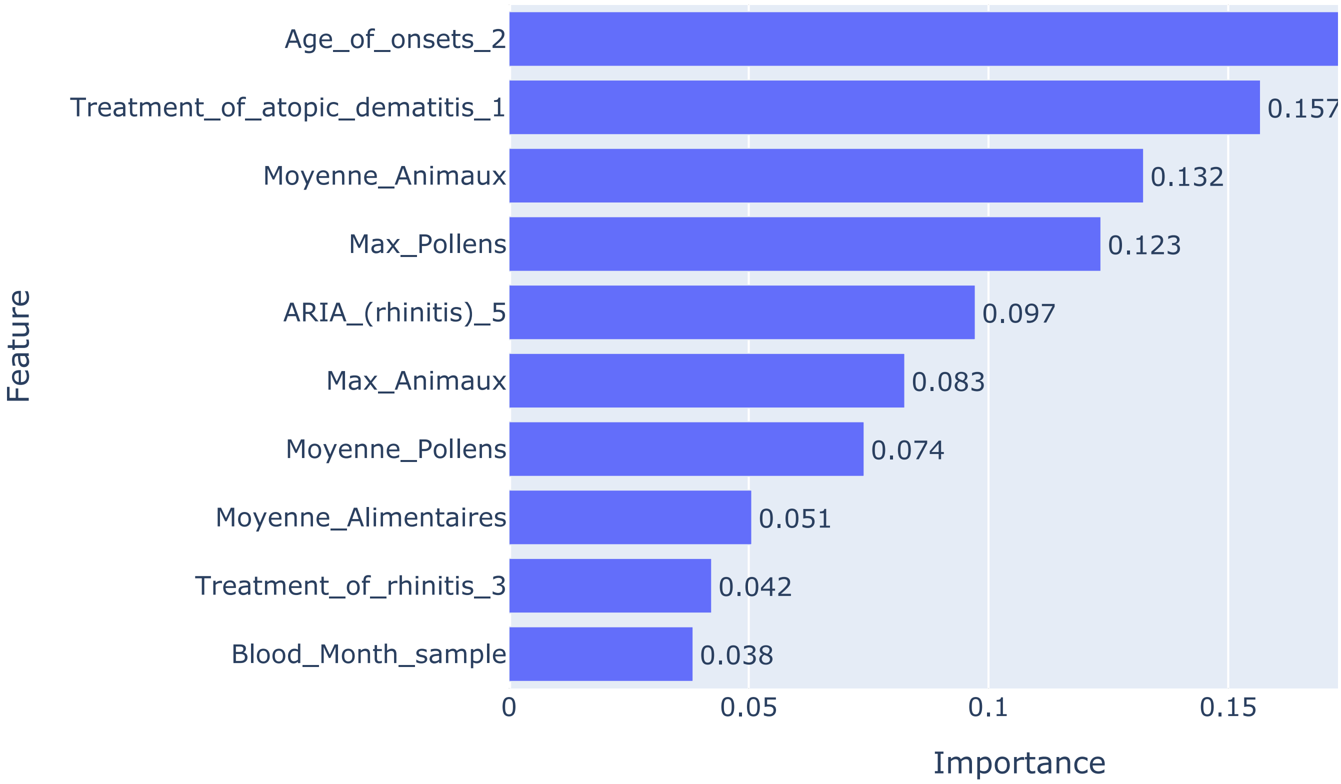
Top 10 Features pour la cible 'Venom_Allergy' (XGBoost)

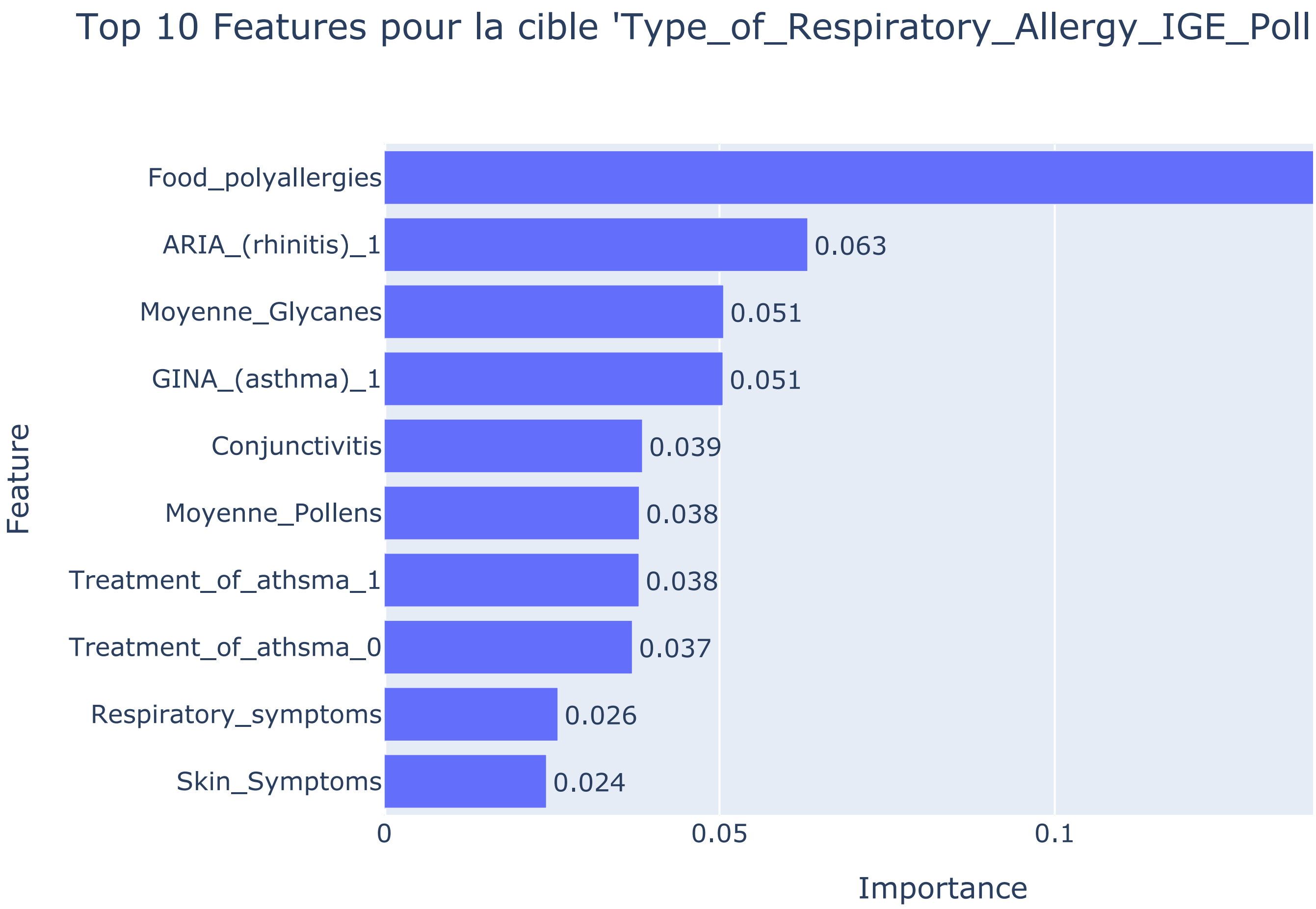


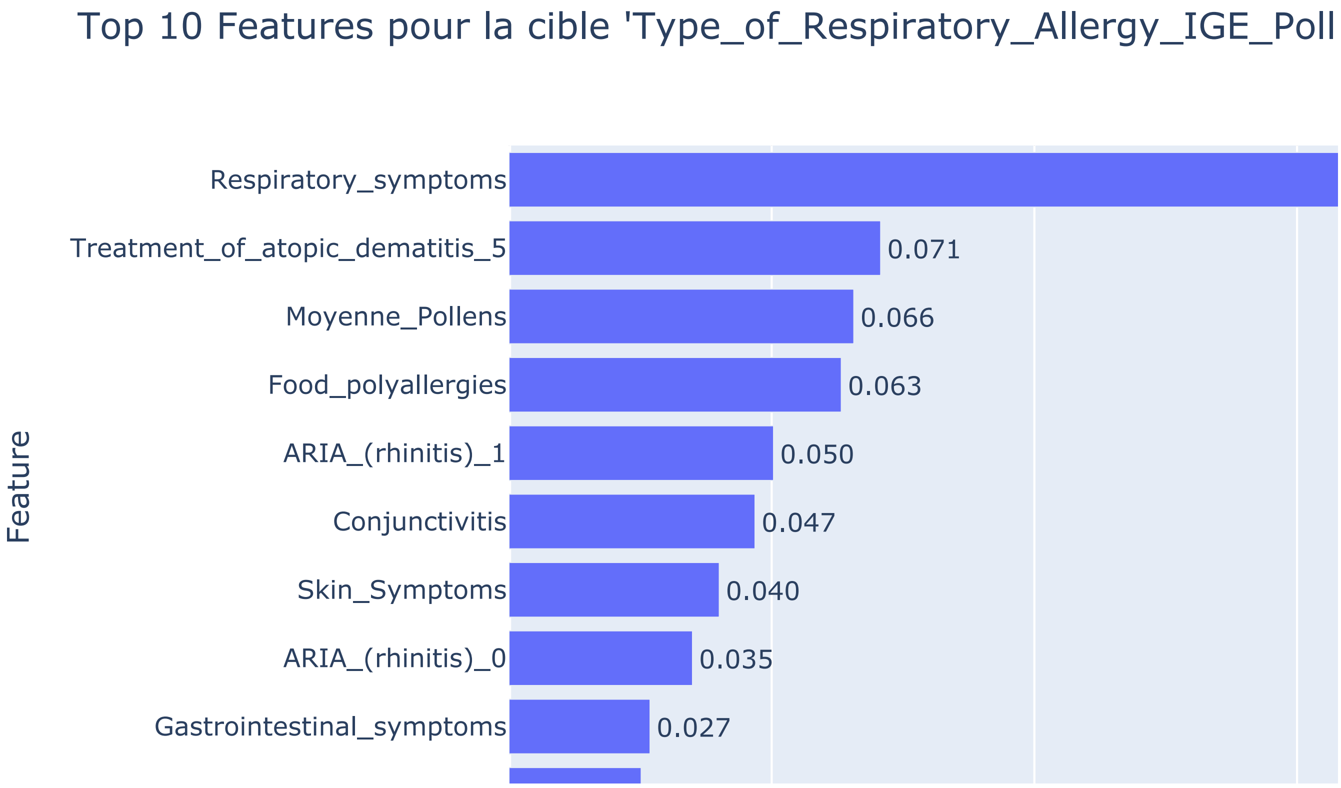
Top 10 Features pour la cible 'Severe_Allergy' (XGBoost)

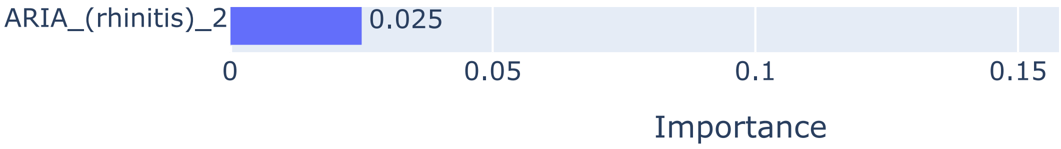


Top 10 Features pour la cible 'Type_of_Food_Allergy_Other' (XGBoos

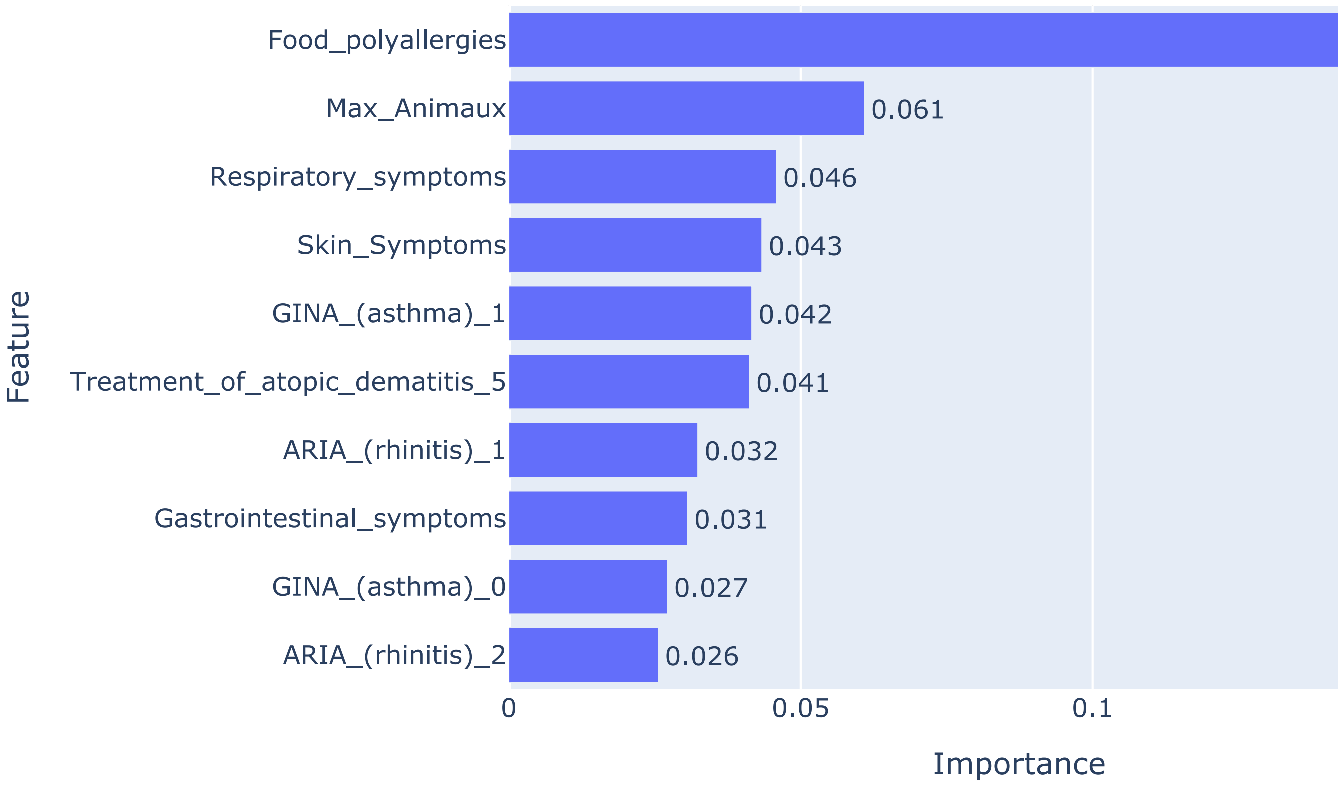



Top 10 Features pour la cible 'Type_of_Respiratory_Allergy_IGE_Poll
Feature
Food_polyallergies
ARIA_(rhinitis)_1
Moyenne_Glycanes
GINA_(asthma)_1
Conjunctivitis
Moyenne_Pollens
Treatment_of_athsma_1
Treatment_of_athsma_0
Respiratory_symptoms
Skin_Symptoms
0.063
0.051
0.051
0.039
0.038
0.038
0.037
0.026
0.024
0
0.05
0.1
Importance

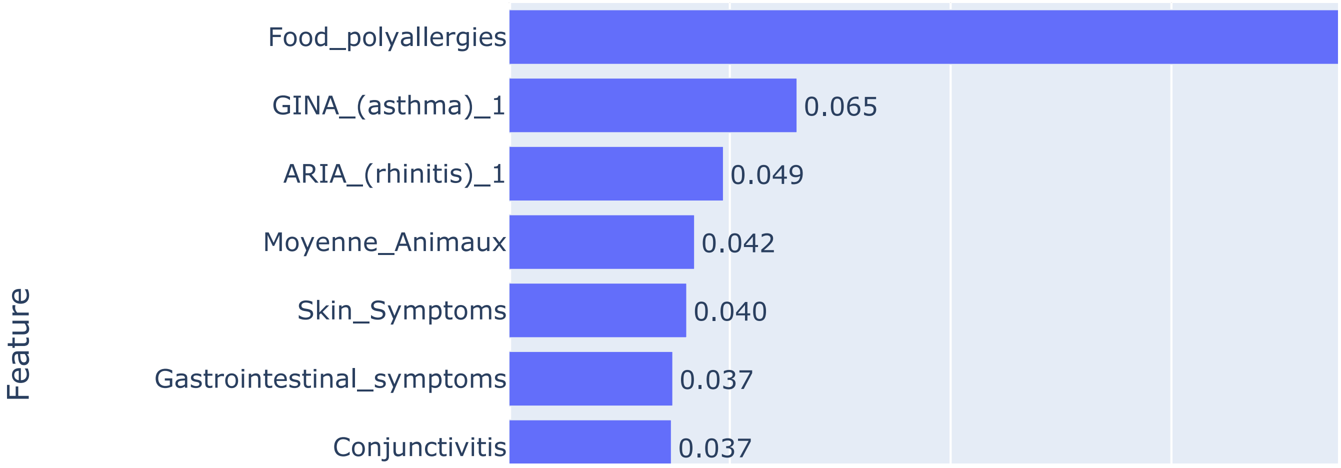

Top 10 Features pour la cible 'Type_of_Respiratory_Allergy_IGE_Poll
Feature
Respiratory_symptoms
Treatment_of_atopic_dematitis_5
Moyenne_Pollens
Food_polyallergies
ARIA_(rhinitis)_1
Conjunctivitis
Skin_Symptoms
ARIA_(rhinitis)_0
Gastrointestinal_symptoms
0.071
0.066
0.063
0.050
0.047
0.040
0.035
0.027

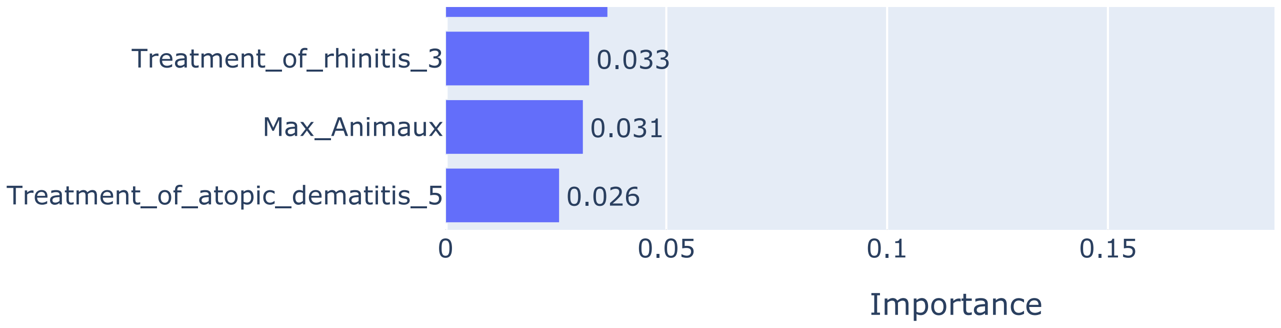


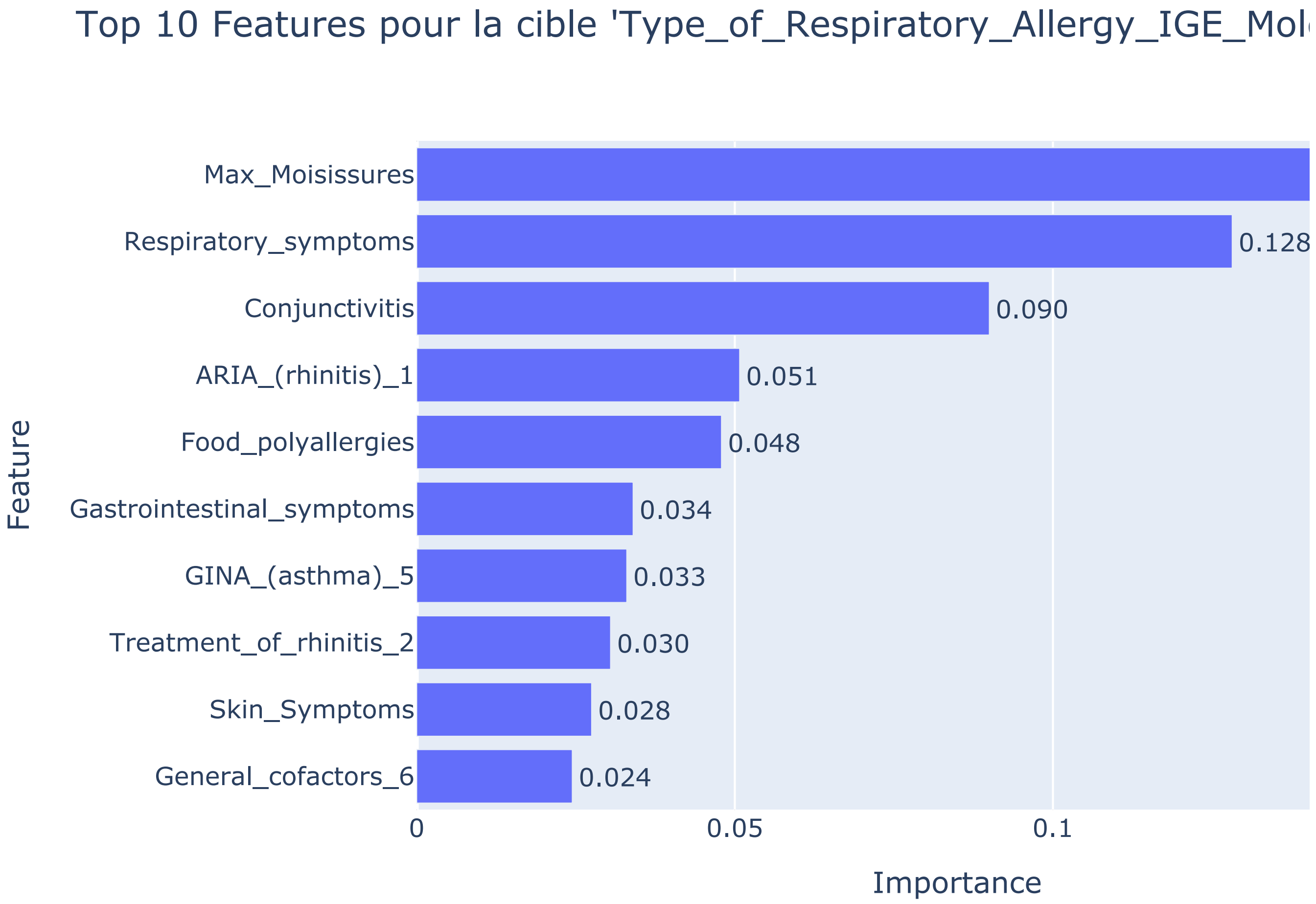
Top 10 Features pour la cible 'Type_of_Respiratory_Allergy_IGE_Dar

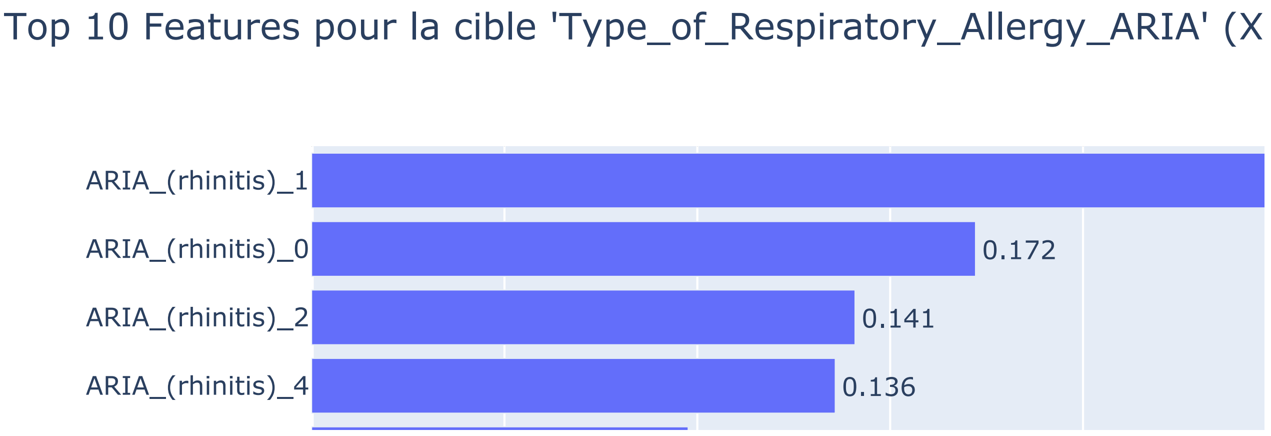


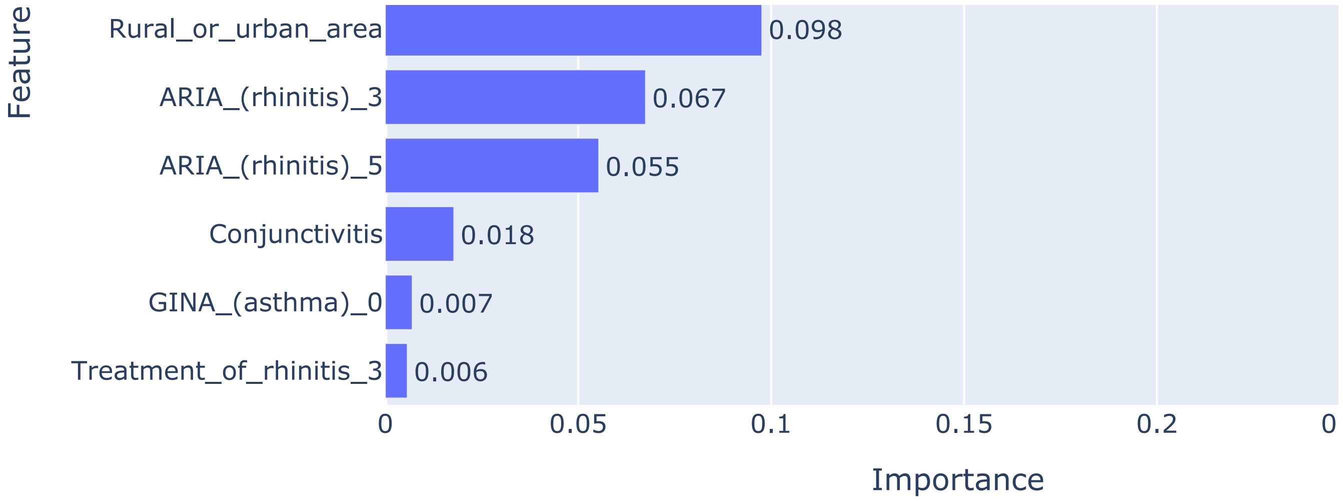
Top 10 Features pour la cible 'Type_of_Respiratory_Allergy_IGE_Mite



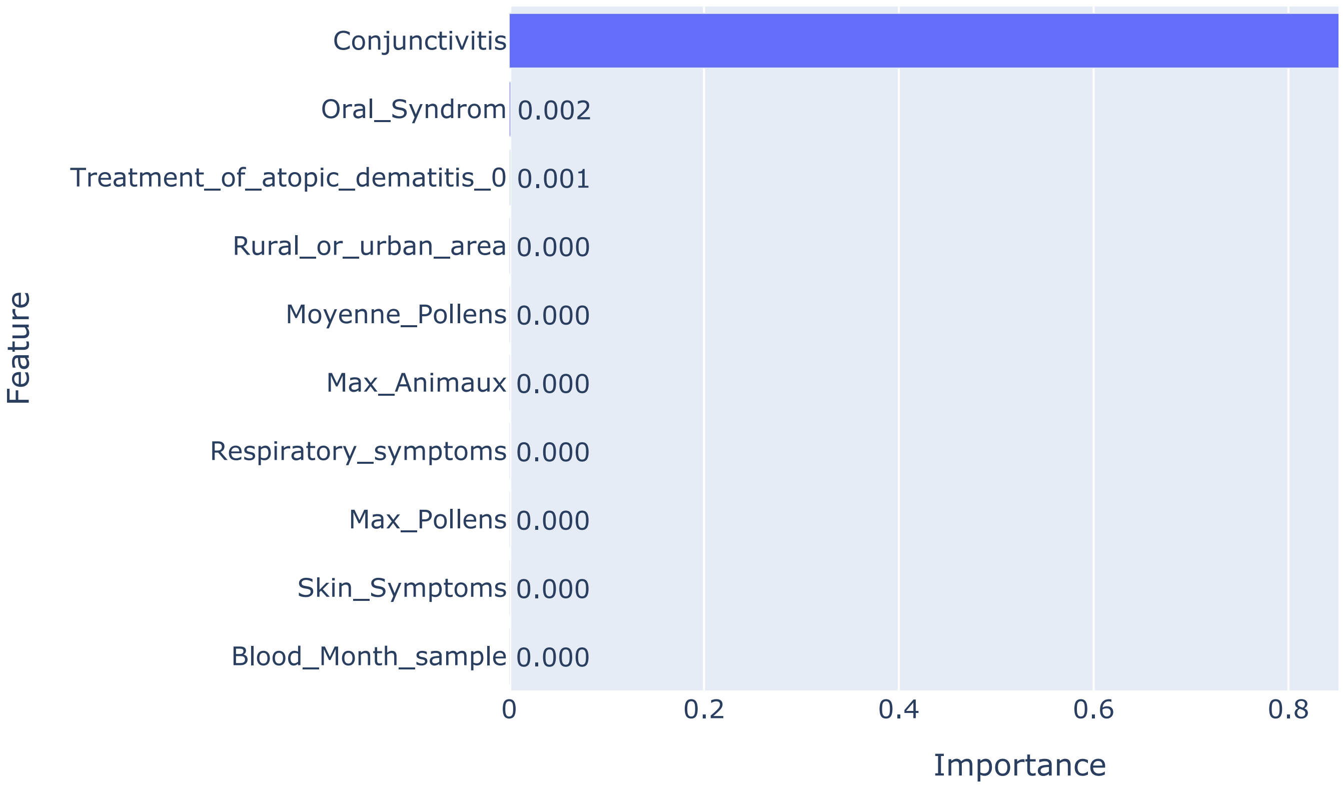



Top 10 Features pour la cible 'Type_of_Respiratory_Allergy_IGE_Mol


Top 10 Features pour la cible 'Type_of_Respiratory_Allergy_ARIA' (X

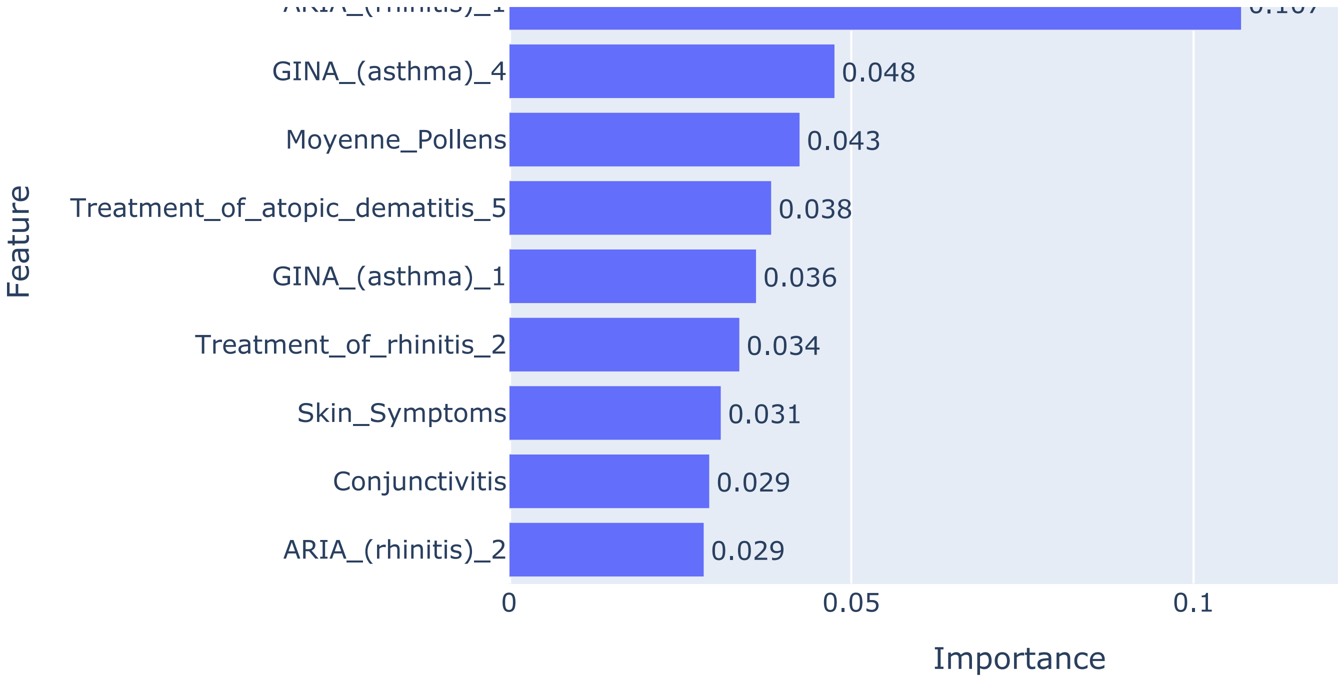


Top 10 Features pour la cible 'Type_of_Respiratory_Allergy_CONJ' (X

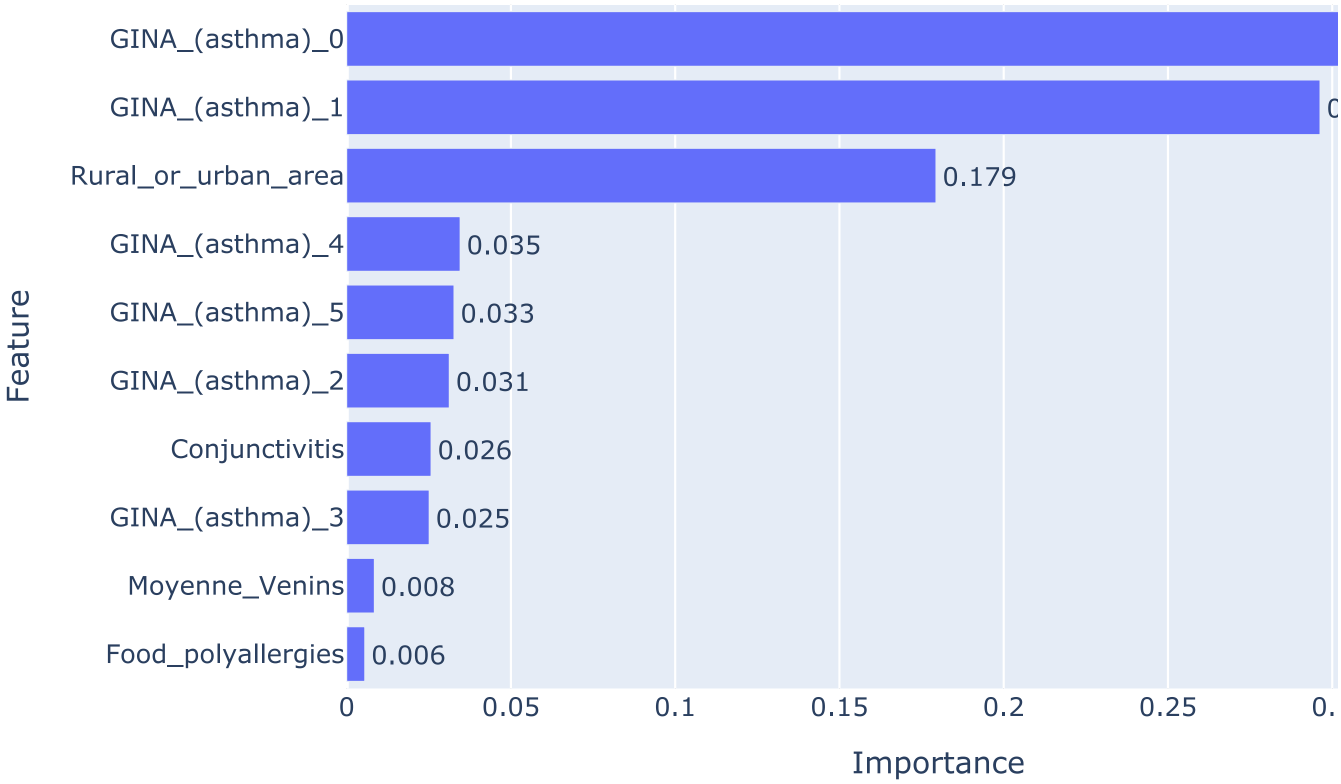


Top 10 Features pour la cible 'Type_of_Respiratory_Allergy_IGE_Poll

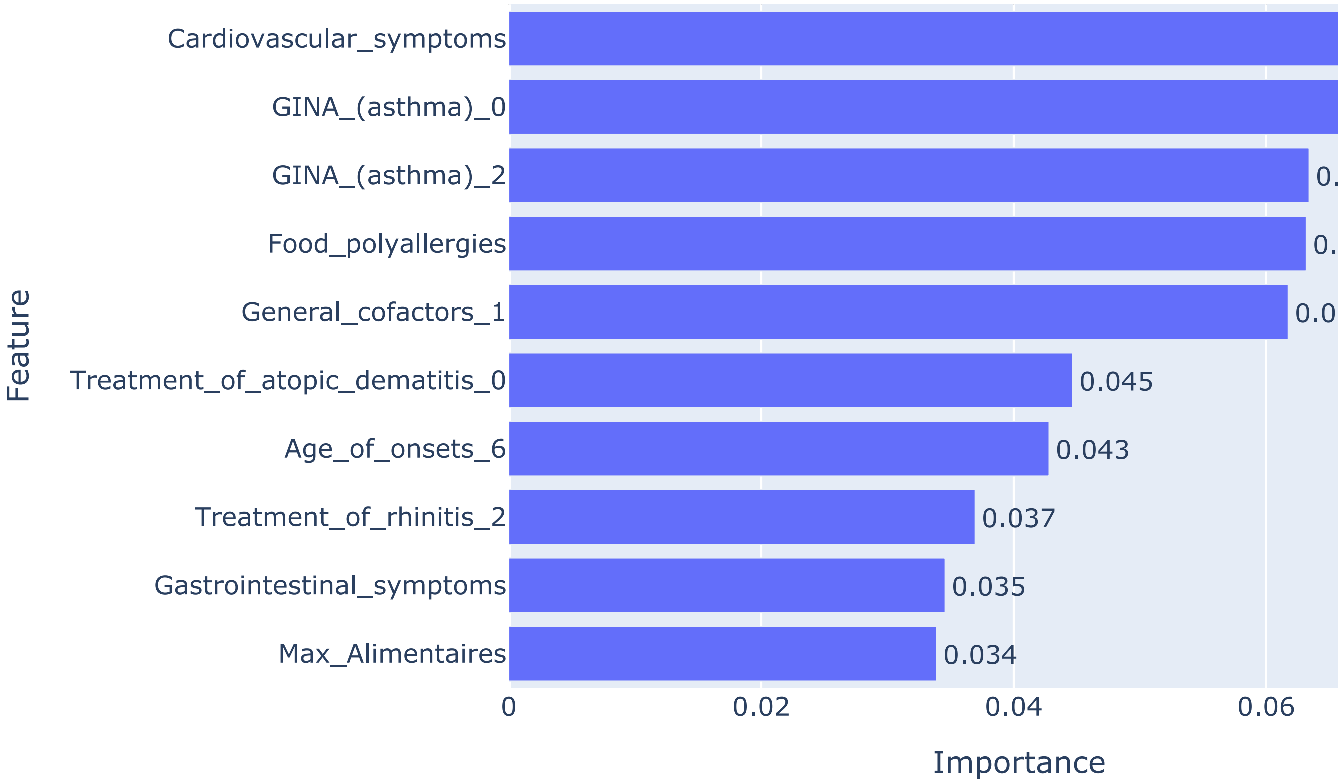
Food_polyallergies
ARIA_(rhinitis)_1 0.107



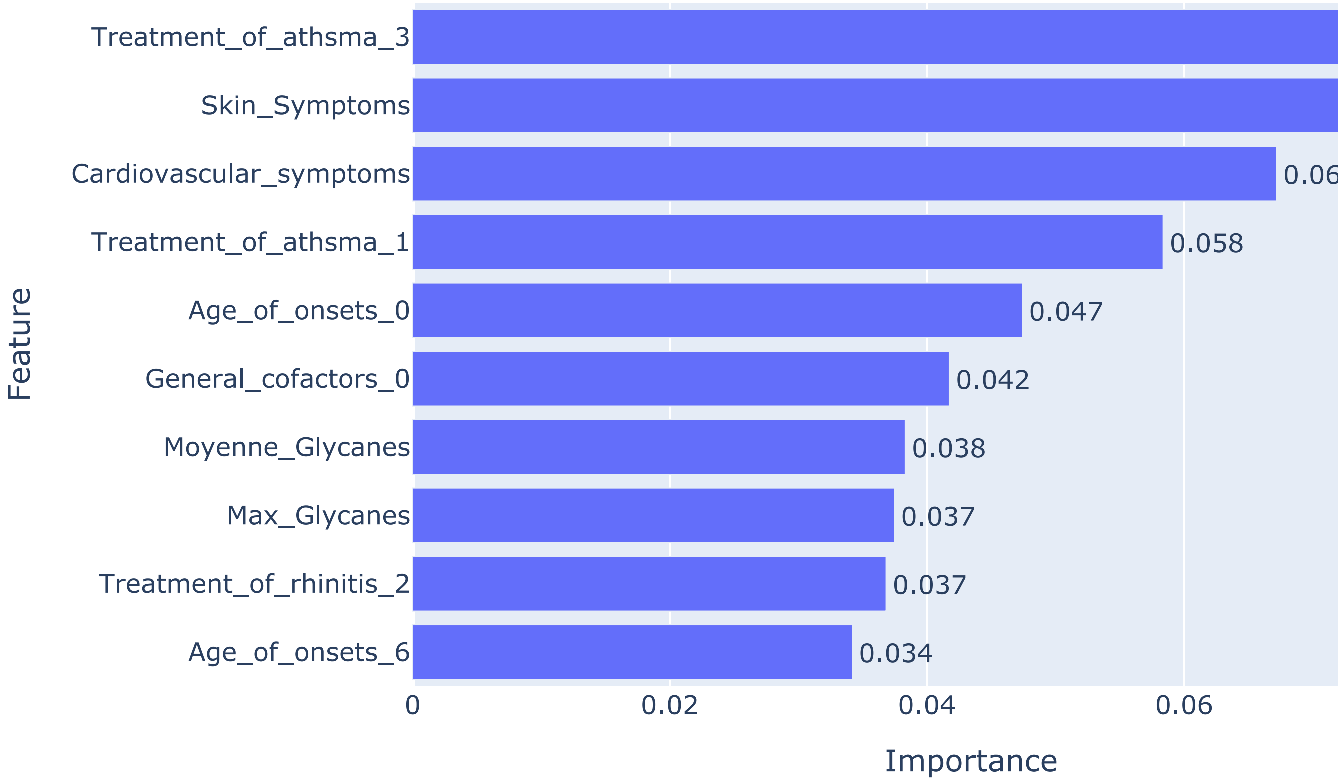
Top 10 Features pour la cible 'Type_of_Respiratory_Allergy_GINA' (X



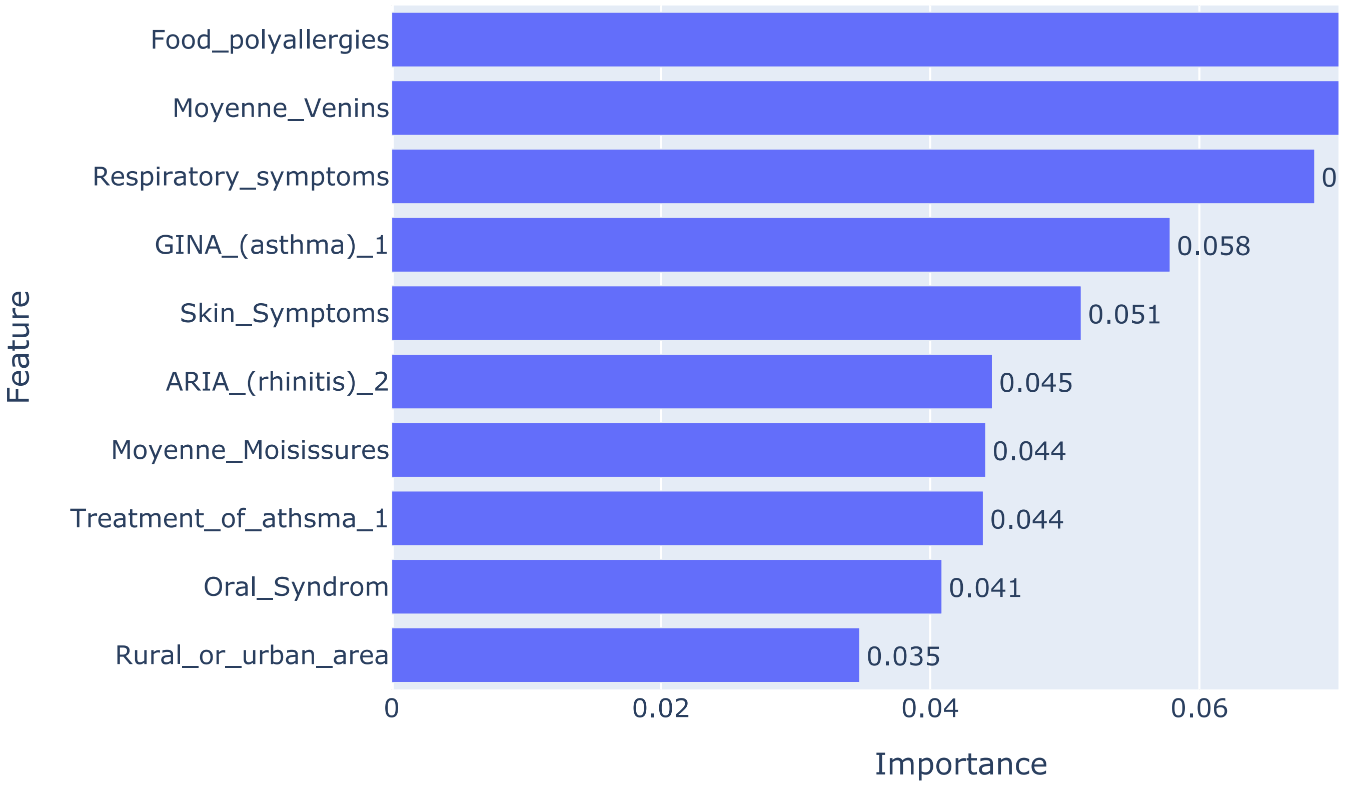
Top 10 Features pour la cible 'Type_of_Food_Allergy_Aromatics' (XG



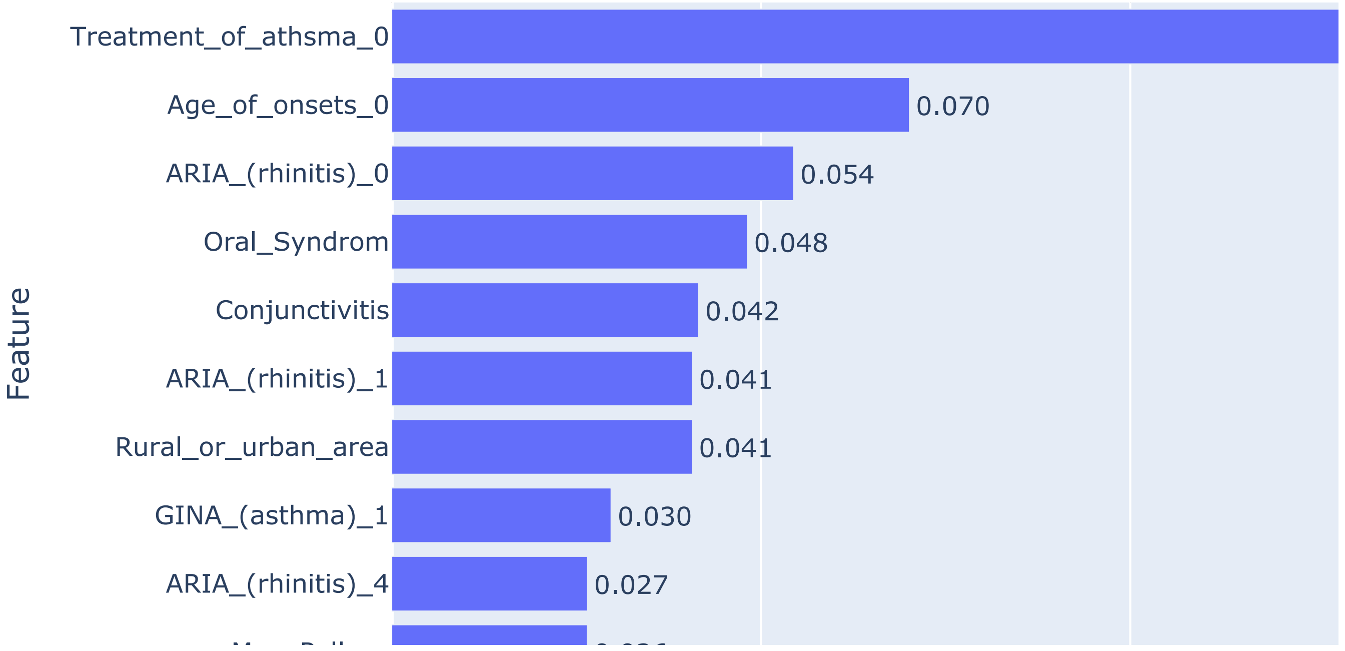
Top 10 Features pour la cible 'Type_of_Food_Allergy_Cereals_&_See

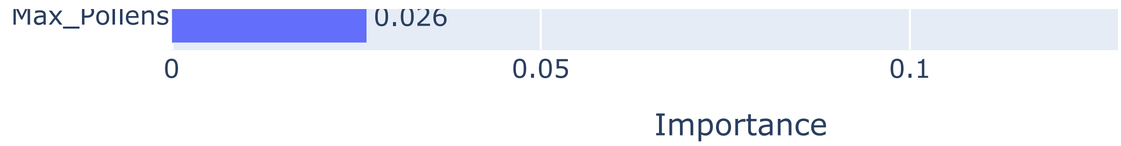


Top 10 Features pour la cible 'Type_of_Food_Allergy_Egg' (XGBoost)

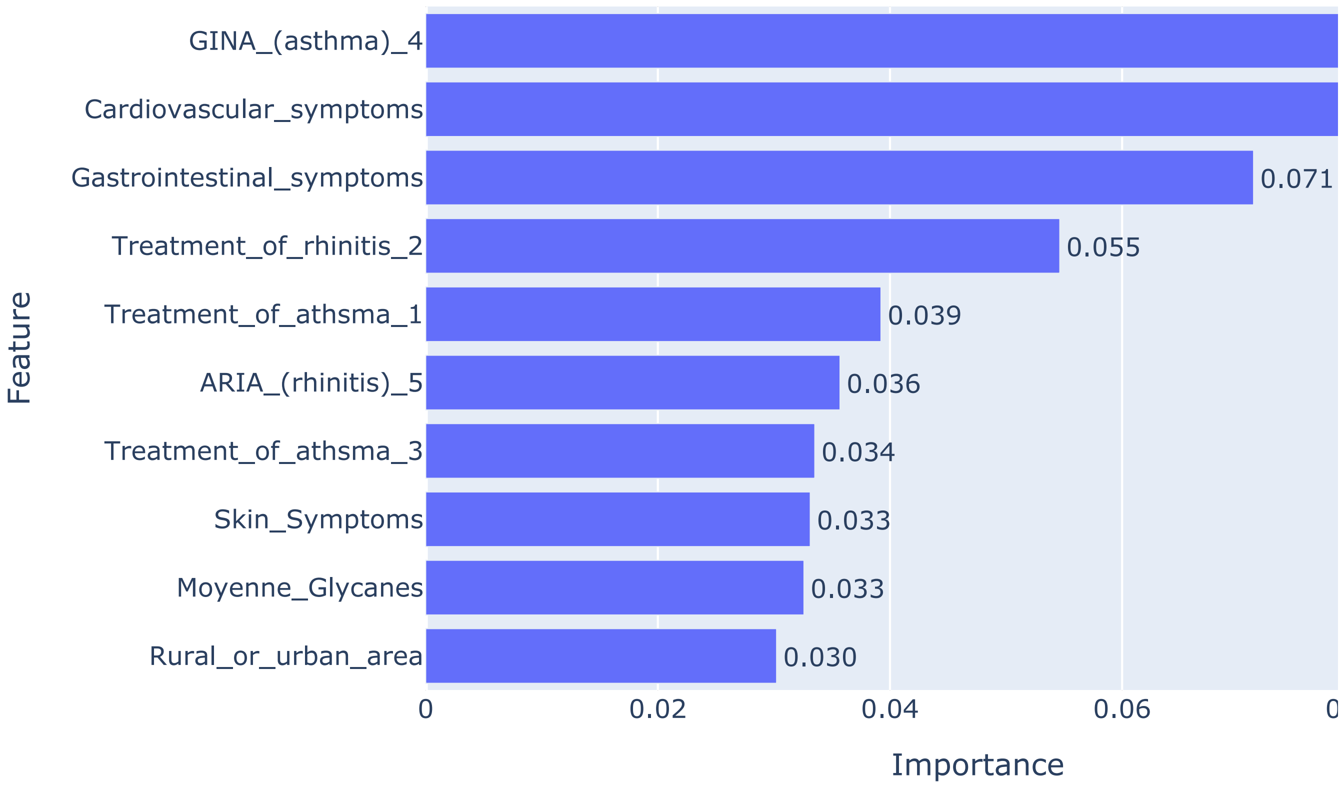


Top 10 Features pour la cible 'Type_of_Food_Allergy_Fish' (XGBoost)

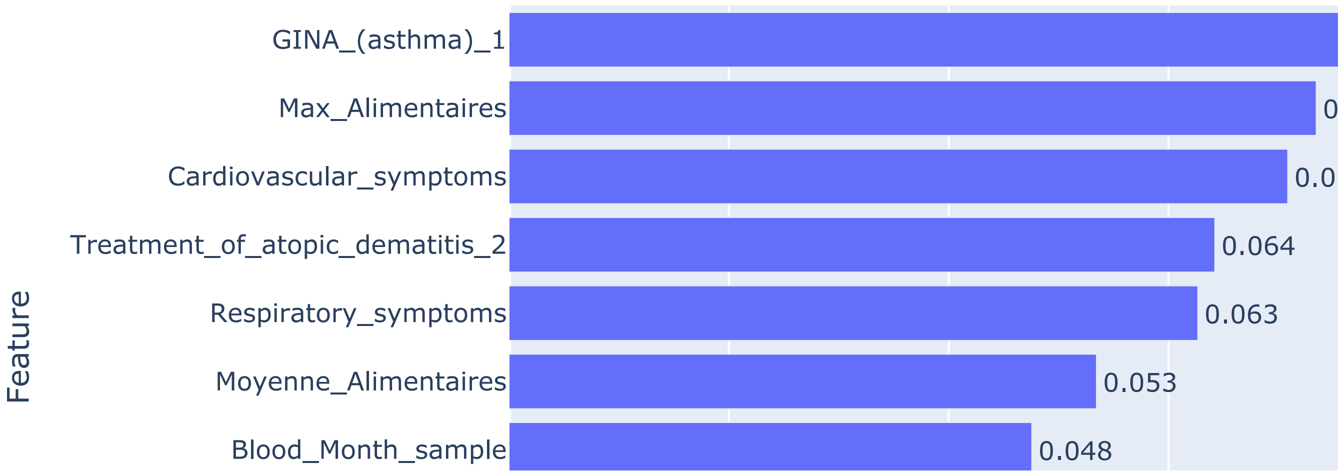


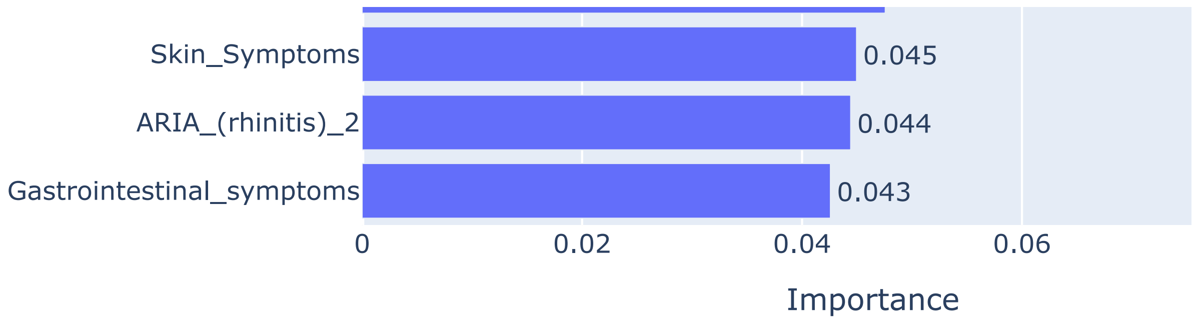


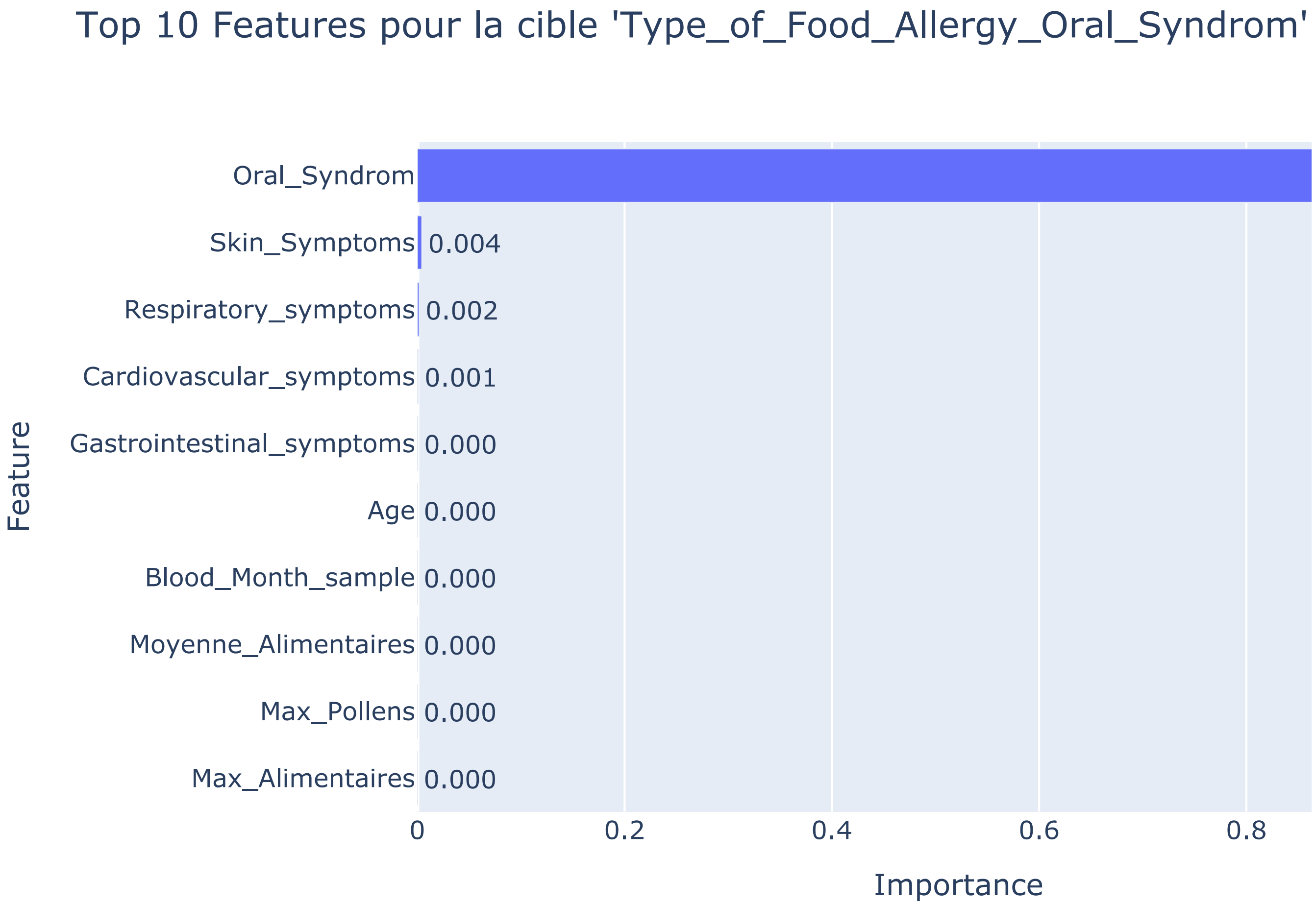
Top 10 Features pour la cible 'Type_of_Food_Allergy_Fruits_and_Veg

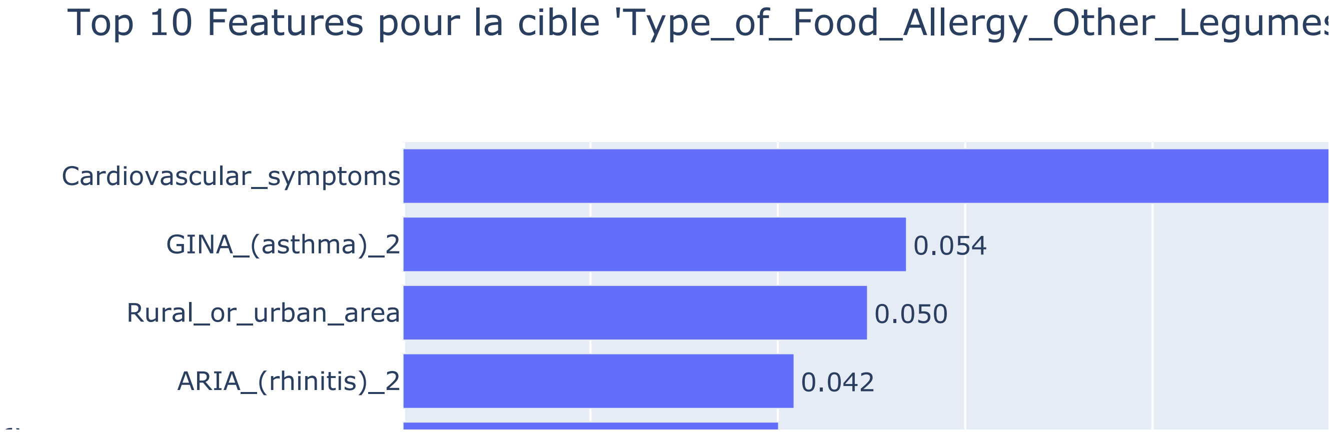


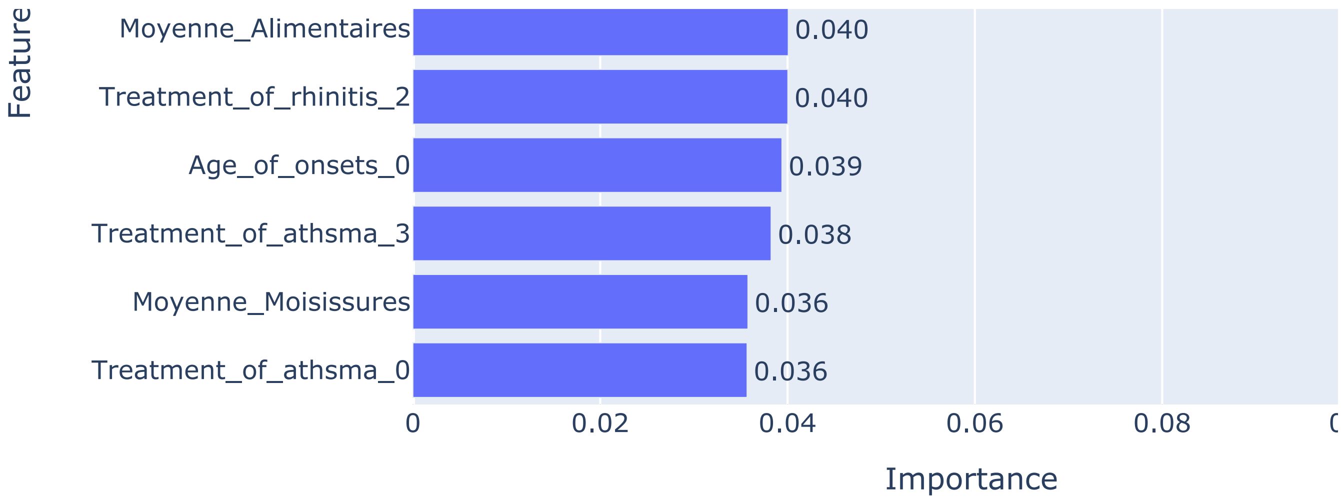
Top 10 Features pour la cible 'Type_of_Food_Allergy_Mammalian_Mi



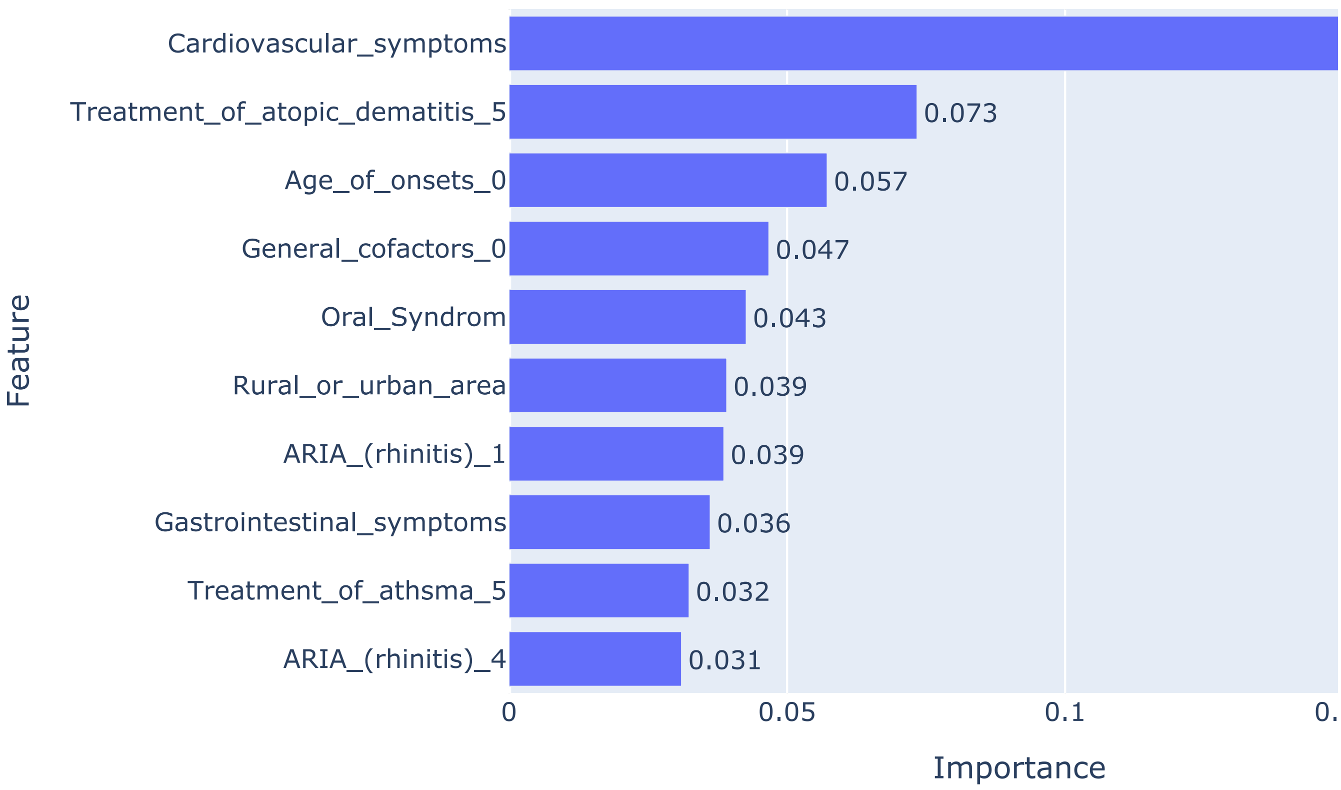



Top 10 Features pour la cible 'Type_of_Food_Allergy_Oral_Syndrom'

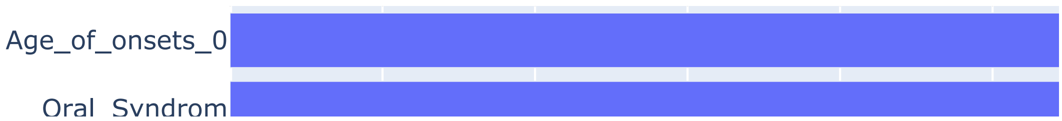

Top 10 Features pour la cible 'Type_of_Food_Allergy_Other_Legume

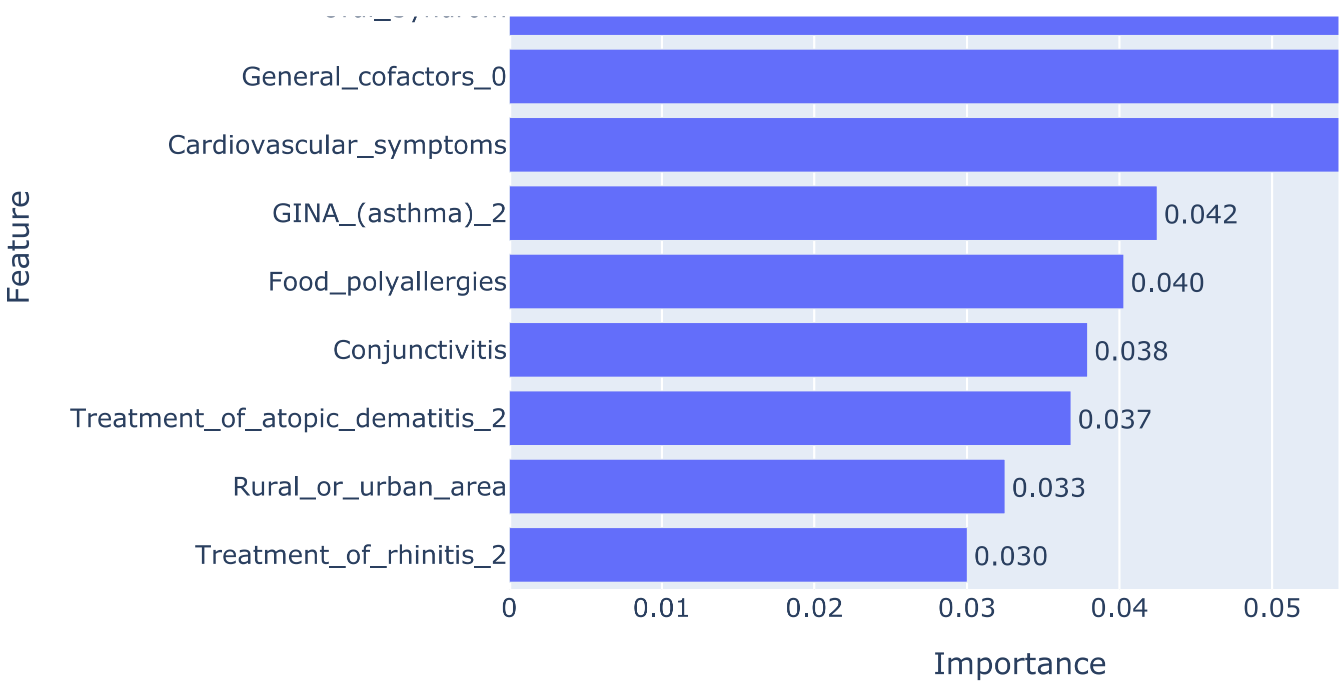


Top 10 Features pour la cible 'Type_of_Food_Allergy_Peanut' (XGBoo

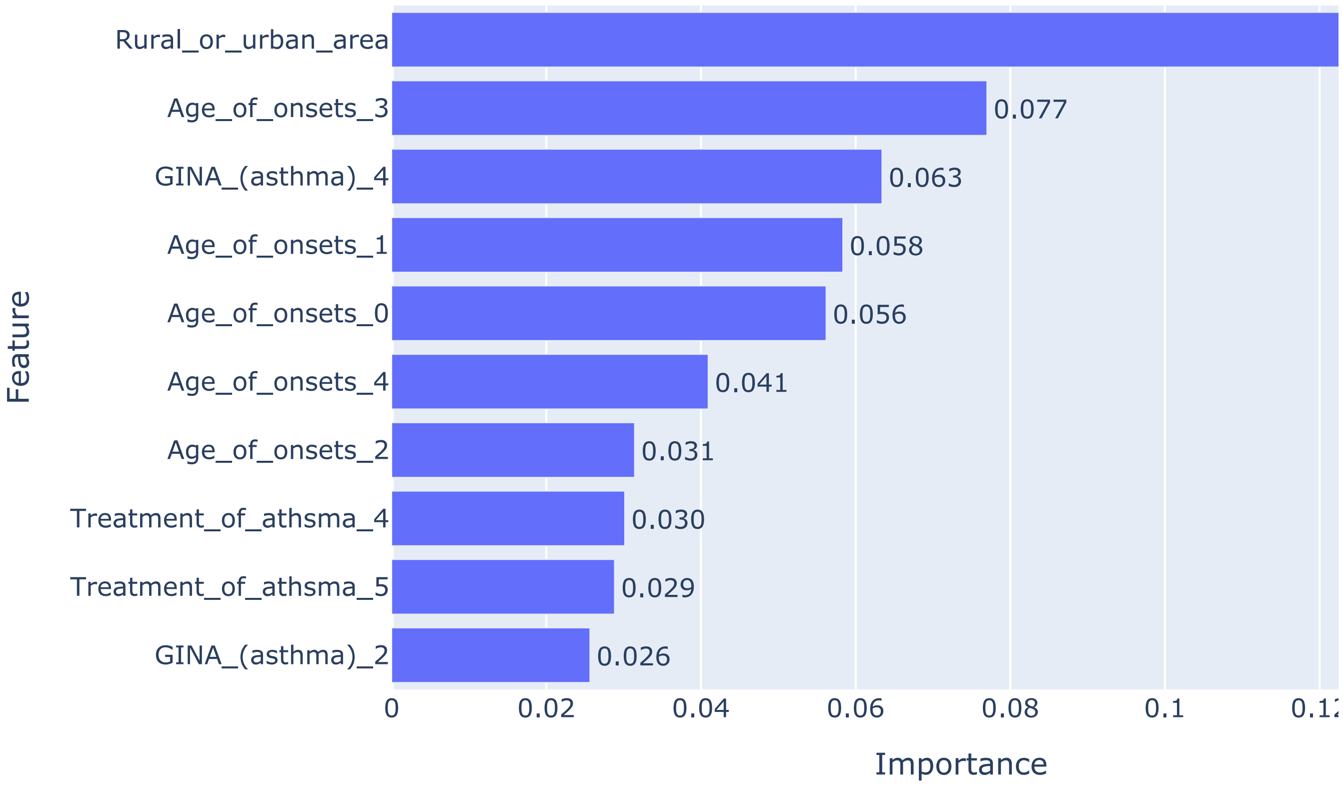


Top 10 Features pour la cible 'Type_of_Food_Allergy_Shellfish' (XGB

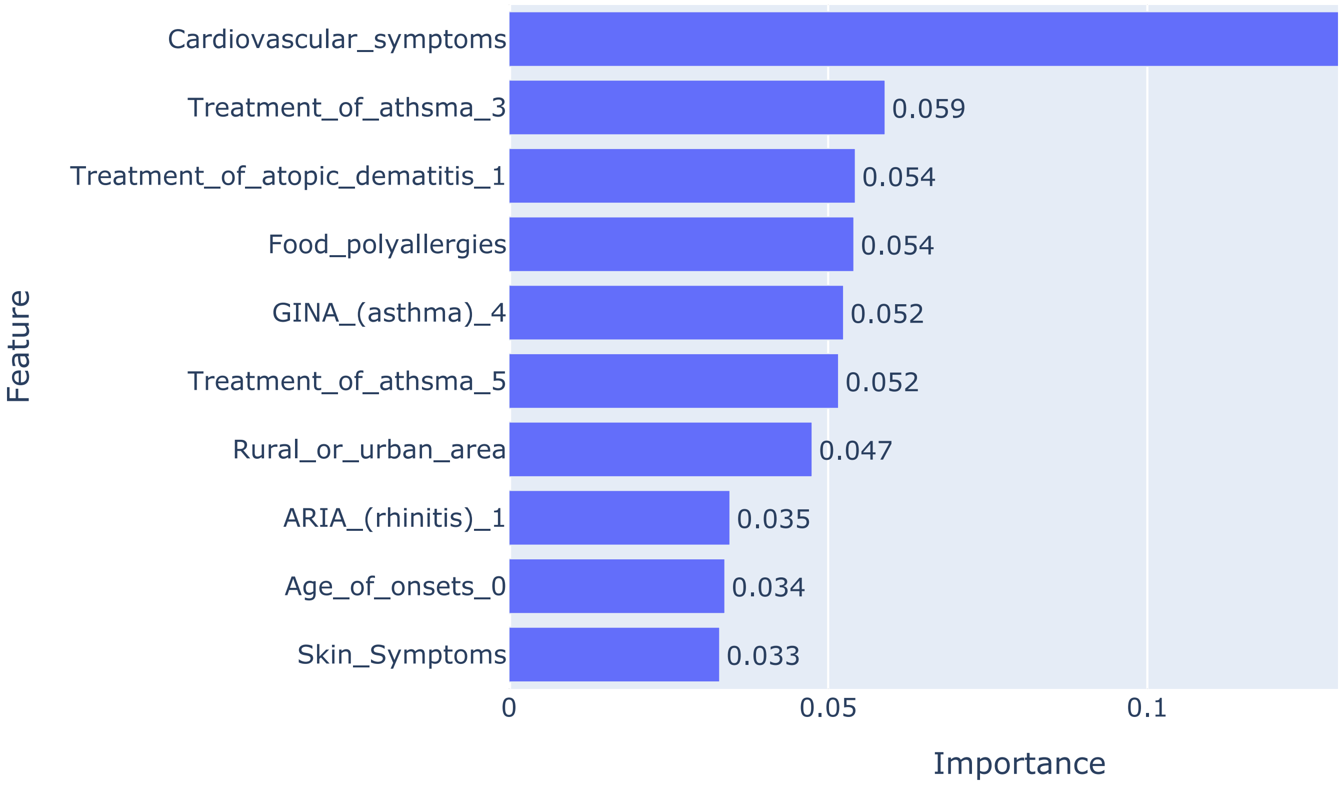




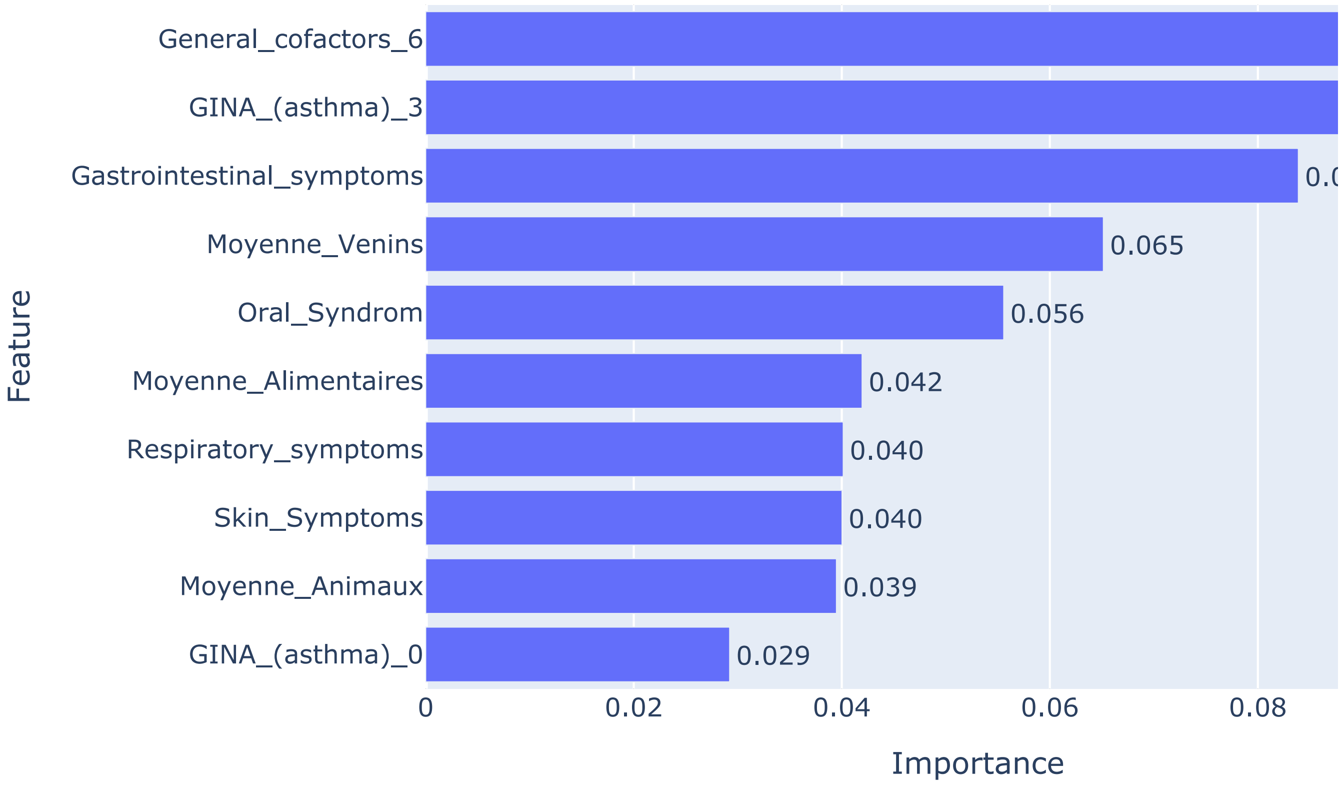
Top 10 Features pour la cible 'Type_of_Food_Allergy_TPO' (XGBoost)

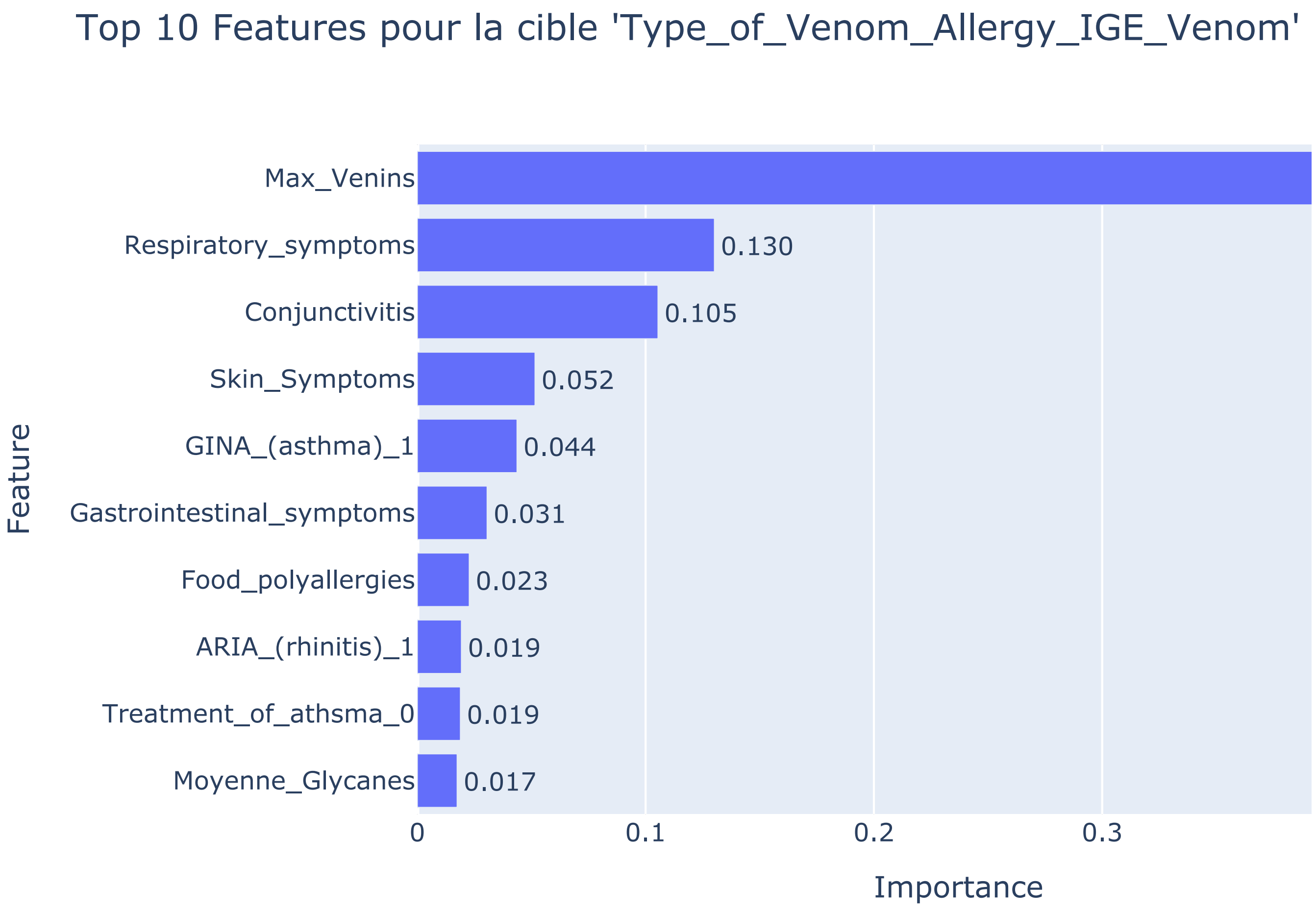


Top 10 Features pour la cible 'Type_of_Food_Allergy_Tree_Nuts' (XG



Top 10 Features pour la cible 'Type_of_Venom_Allergy_ATCD_Venom




Top 10 Features pour la cible 'Type_of_Venom_Allergy_IGE_Venom'
Feature
Max_Venins
Respiratory_symptoms
0.130
Conjunctivitis
0.105
Skin_Symptoms
0.052
GINA_(asthma)_1
0.044
Gastrointestinal_symptoms
0.031
Food_polyallergies
0.023
ARIA_(rhinitis)_1
0.019
Treatment_of_athsma_0
0.019
Moyenne_Glycanes
0.017
0
0.1
0.2
0.3
Importance

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.