random_forest

July 29, 2025

[2]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score,
    f1_score, roc_auc_score, confusion_matrix, roc_curve
)
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
```

[3]:
```
df = pd.read_parquet("/PHI_conf/VaccineUptake/Analysts/Vay/
 ↪vaccinate_uptake_ML_analysis/vaccine_data/master_data/cohort_df_merged.
 ↪parquet")
unique_cohorts = df['cohort_group_ML_analysis'].unique()
unique_cohorts
```

[3]:
```
array(['AGE_50_TO_64', 'AGE_65_TO_74', 'AGE_75_AND_OVER',
       'ALL_HEALTH_CARE_WORKERS', 'ALL_SOCIAL_CARE_WORKERS',
       '18_TO_64_FLU_AT_RISK', 'OLDER_PEOPLE_CARE_HOME',
       'WEAKENED_IMMUNE_SYSTEM'], dtype=object)
```

[4]:
```
unique_cohorts = df['cohort_group_ML_analysis'].unique()
unique_cohorts
```

[4]:
```
array(['AGE_50_TO_64', 'AGE_65_TO_74', 'AGE_75_AND_OVER',
       'ALL_HEALTH_CARE_WORKERS', 'ALL_SOCIAL_CARE_WORKERS',
       '18_TO_64_FLU_AT_RISK', 'OLDER_PEOPLE_CARE_HOME',
       'WEAKENED_IMMUNE_SYSTEM'], dtype=object)
```

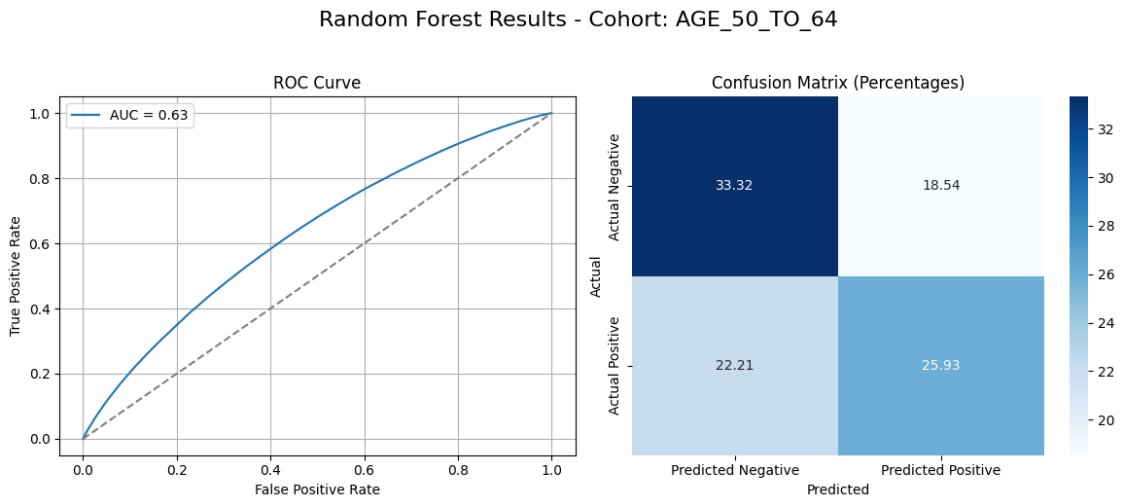
[5]:
```
rf_cross_cohort_summary = {}
rf_model_scores = {}

for cohort in unique_cohorts:
    print(f"\n\n======== Cohort: {cohort} =========")
    cohort_df = df[df["cohort_group_ML_analysis"] == cohort].copy()
```

```
    # Prepare features and target
    X = cohort_df.drop(columns=["cohort_group_ML_analysis",␣
↪"attended_vaccination_event"])
    y = cohort_df["attended_vaccination_event"]

    # Train-test split
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.3, random_state=42, stratify=y
    )
    print("Train size:", len(X_train))
    print("Test size:", len(X_test))

    # Fit Random Forest
    rf_model = RandomForestClassifier(n_estimators=100, random_state=42,␣
↪n_jobs=-1)
    rf_model.fit(X_train, y_train)

    # Predict
    y_prob = rf_model.predict_proba(X_test)[:, 1]
    y_pred = rf_model.predict(X_test)

    # ROC & AUC
    fpr, tpr, _ = roc_curve(y_test, y_prob)
    auc_score = roc_auc_score(y_test, y_prob)

    # Metrics summary
    metrics_summary = pd.DataFrame({
        "Metric": ["Accuracy", "Precision", "Recall", "F1 Score", "AUC Score"],
        "Value": [
            accuracy_score(y_test, y_pred),
            precision_score(y_test, y_pred),
            recall_score(y_test, y_pred),
            f1_score(y_test, y_pred),
            auc_score
        ]
    })

    rf_model_scores[cohort] = {
        "F1 Score": f1_score(y_test, y_pred),
        "AUC Score": auc_score
    }

    print("\nModel Performance Metrics:")
    print(metrics_summary.to_string(index=False))

    # Plot ROC + Confusion Matrix
    fig, axes = plt.subplots(1, 2, figsize=(12, 5))
```

```
    # ROC Curve
    axes[0].plot(fpr, tpr, label=f"AUC = {auc_score:.2f}")
    axes[0].plot([0, 1], [0, 1], linestyle="--", color="gray")
    axes[0].set_xlabel("False Positive Rate")
    axes[0].set_ylabel("True Positive Rate")
    axes[0].set_title("ROC Curve")
    axes[0].legend()
    axes[0].grid(True)

    # Confusion Matrix
    cm = confusion_matrix(y_test, y_pred)
    cm_percent = cm.astype('float') / cm.sum() * 100
    sns.heatmap(cm_percent, annot=True, fmt='.2f', cmap='Blues',
                xticklabels=['Predicted Negative', 'Predicted Positive'],
                yticklabels=['Actual Negative', 'Actual Positive'],
                ax=axes[1])
    axes[1].set_xlabel('Predicted')
    axes[1].set_ylabel('Actual')
    axes[1].set_title('Confusion Matrix (Percentages)')

    fig.suptitle(f"Random Forest Results - Cohort: {cohort}", fontsize=16, y=1.
↪03)
    plt.tight_layout()
    plt.show()

    # Feature Importances
    importances = rf_model.feature_importances_
    feature_names = X.columns
    importance_df = pd.DataFrame({
        'Feature': feature_names,
        'Importance': importances
    }).sort_values(by='Importance', ascending=False)

    print("\nTop Feature Importances:")
    print(importance_df.round(4).to_string(index=False))

    # Bar Chart of Feature Importances
    plt.figure(figsize=(8, 4))
    sns.barplot(
        data=importance_df,
        x="Importance",
        y="Feature",
        palette="viridis"
    )
    plt.title(f"Feature Importances - Cohort: {cohort}")
    plt.xlabel("Importance Score")
```

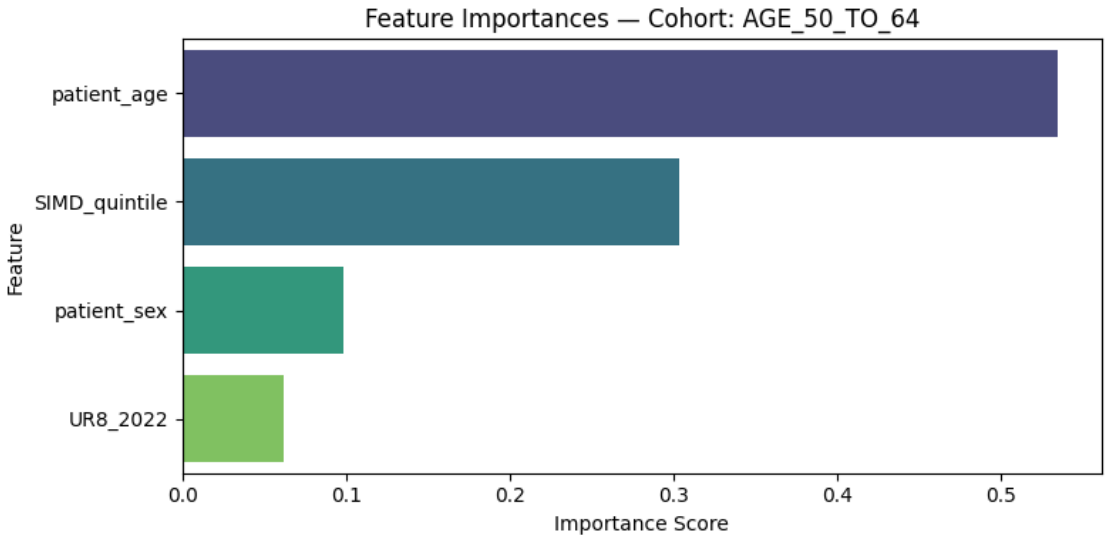
```
    plt.ylabel("Feature")
    plt.tight_layout()
    plt.show()

    # Store results
    rf_cross_cohort_summary[cohort] = dict(zip(
        importance_df['Feature'],
        (importance_df['Importance'] * 100).round(1).astype(str) + "%"
    ))
```

```
======== Cohort: AGE_50_TO_64 =========
Train size: 2068840
Test size: 886646

Model Performance Metrics:
   Metric    Value
 Accuracy 0.592489
Precision 0.583055
   Recall 0.538573
 F1 Score 0.559932
AUC Score 0.628166
```

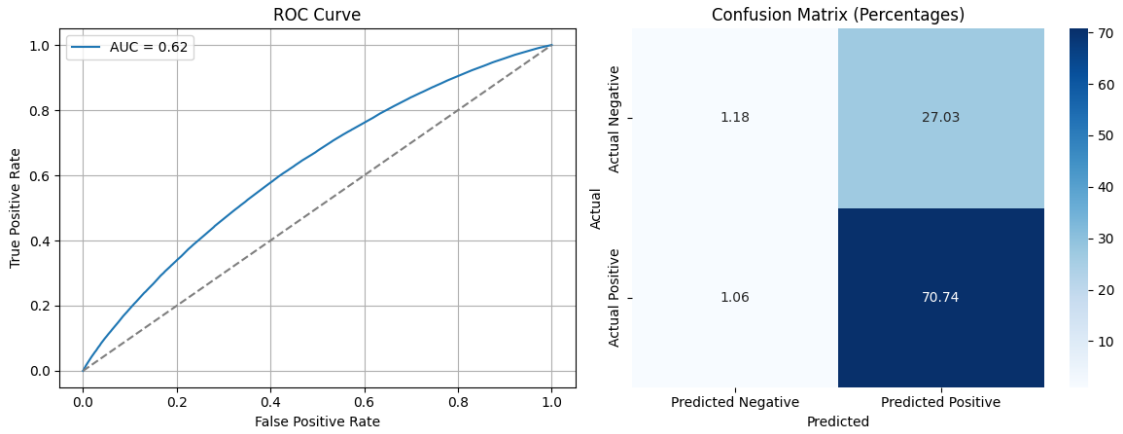



```
Top Feature Importances:
      Feature  Importance
  patient_age      0.5352
SIMD_quintile      0.3038
  patient_sex      0.0988
```

```
     UR8_2022      0.0622
/tmp/ipykernel_2887/1701446235.py:91: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(
```

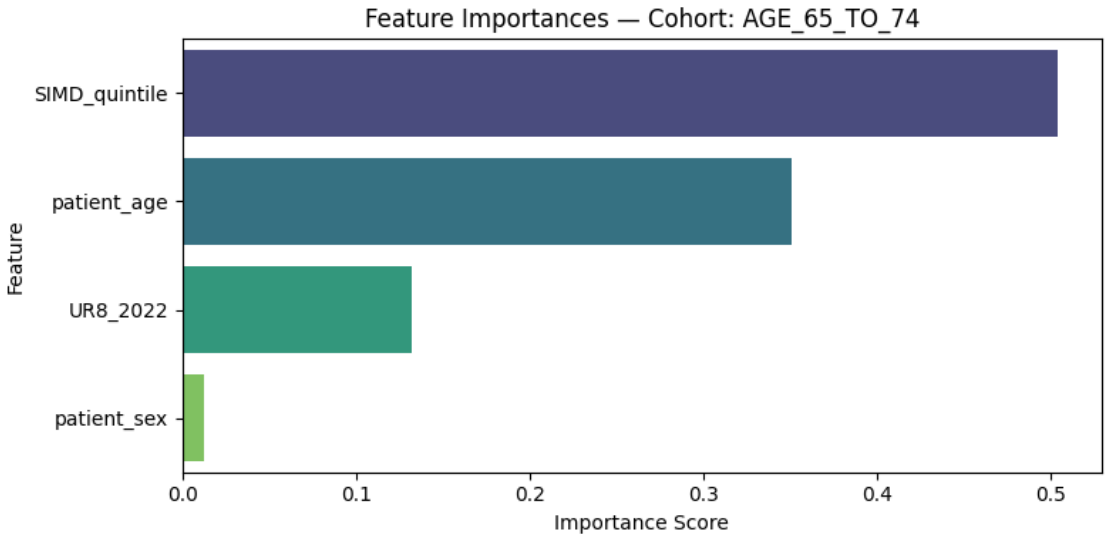



```
======== Cohort: AGE_65_TO_74 =========
Train size: 896325
Test size: 384140

Model Performance Metrics:
   Metric    Value
 Accuracy 0.719128
Precision 0.723515
   Recall 0.985293
 F1 Score 0.834352
AUC Score 0.623319
```

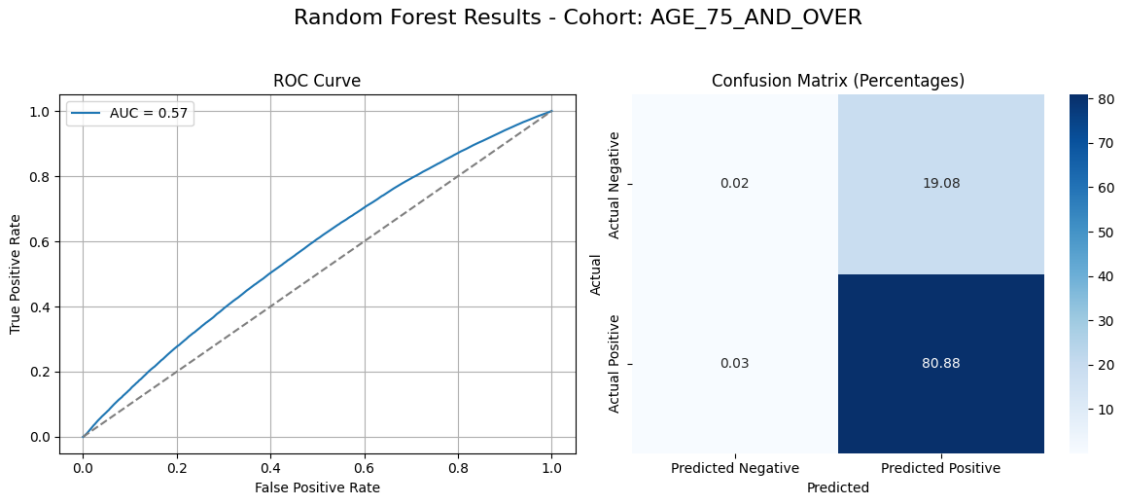



```
Top Feature Importances:
      Feature  Importance
SIMD_quintile      0.5045
  patient_age      0.3507
     UR8_2022      0.1323
  patient_sex      0.0125
```

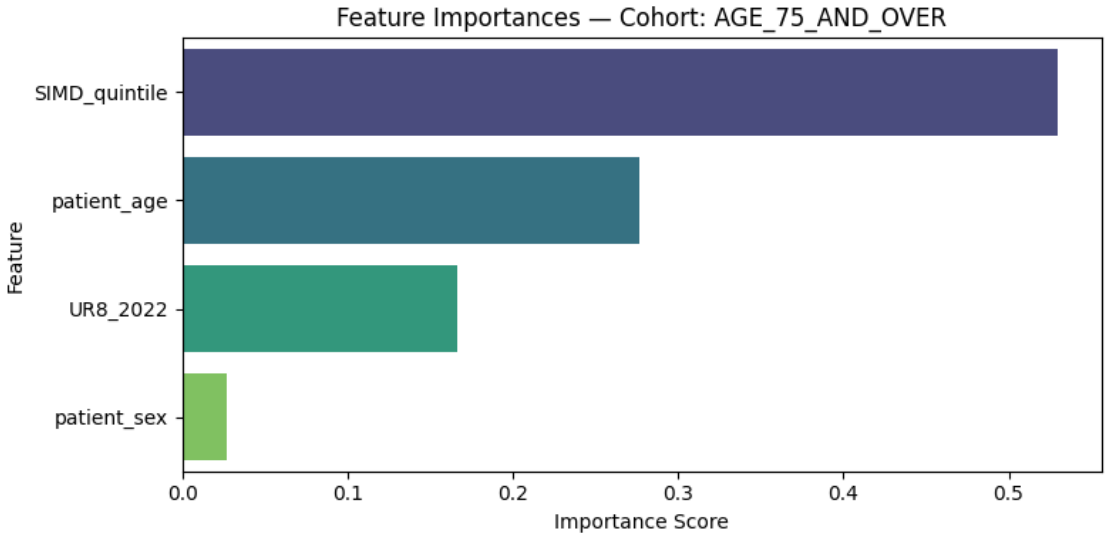
```
/tmp/ipykernel_2887/1701446235.py:91: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(
```



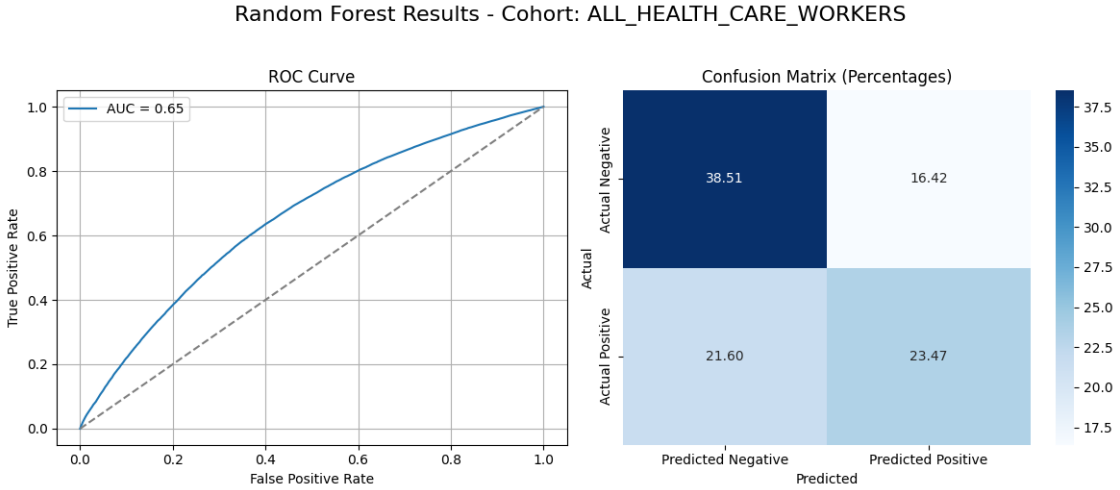
```
======== Cohort: AGE_75_AND_OVER =========
Train size: 794668
Test size: 340572

Model Performance Metrics:
   Metric    Value
 Accuracy 0.808930
Precision 0.809152
   Recall 0.999615
 F1 Score 0.894356
AUC Score 0.574187
```




```
Top Feature Importances:
      Feature  Importance
SIMD_quintile      0.5299
  patient_age      0.2766
     UR8_2022      0.1667
  patient_sex      0.0268
/tmp/ipykernel_2887/1701446235.py:91: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(
```

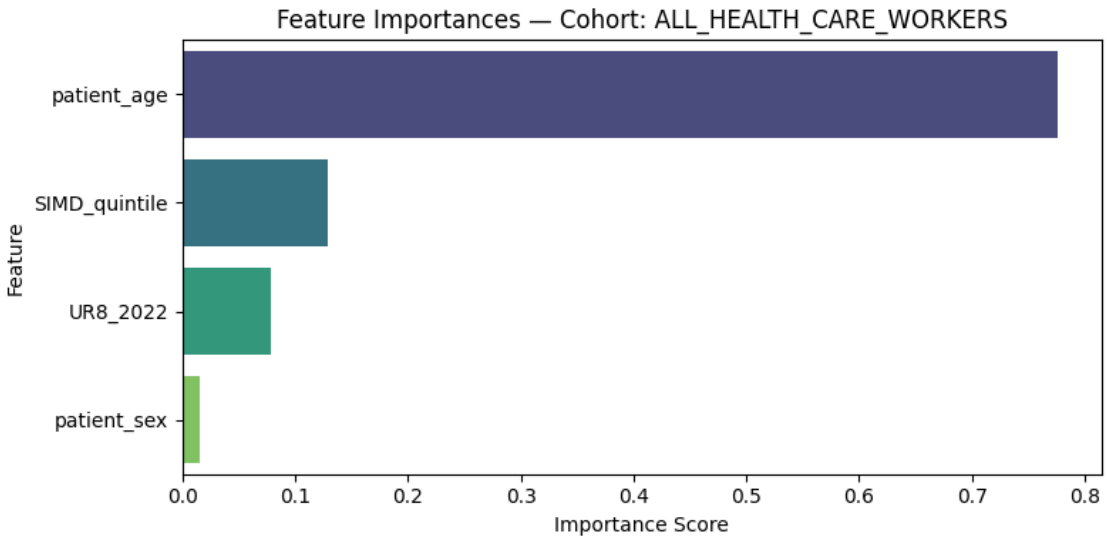



```
======== Cohort: ALL_HEALTH_CARE_WORKERS =========
Train size: 317592
Test size: 136111

Model Performance Metrics:
   Metric    Value
 Accuracy 0.619862
Precision 0.588452
   Recall 0.520816
 F1 Score 0.552572
AUC Score 0.654638
```



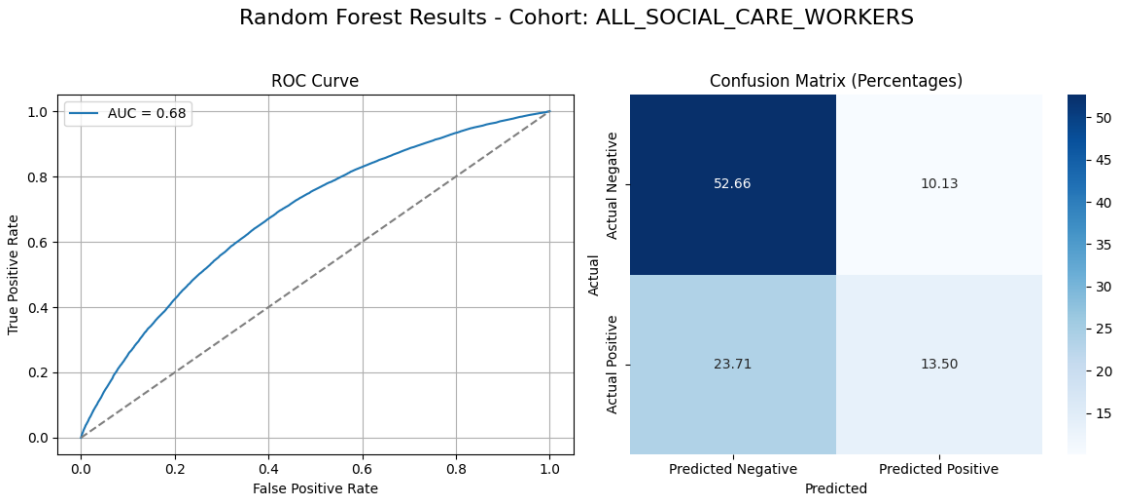
```
Top Feature Importances:
      Feature  Importance
  patient_age      0.7764
SIMD_quintile      0.1291
     UR8_2022      0.0785
  patient_sex      0.0159
```

```
/tmp/ipykernel_2887/1701446235.py:91: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(
```

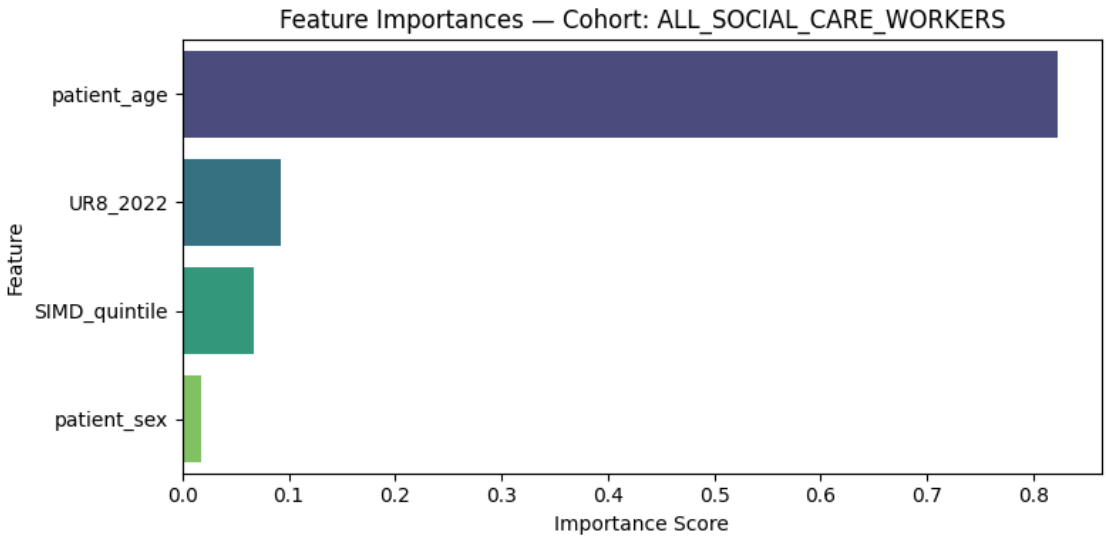



```
======== Cohort: ALL_SOCIAL_CARE_WORKERS =========
Train size: 201854
Test size: 86509

Model Performance Metrics:
   Metric    Value
 Accuracy 0.661631
Precision 0.571345
   Recall 0.362857
 F1 Score 0.443836
```

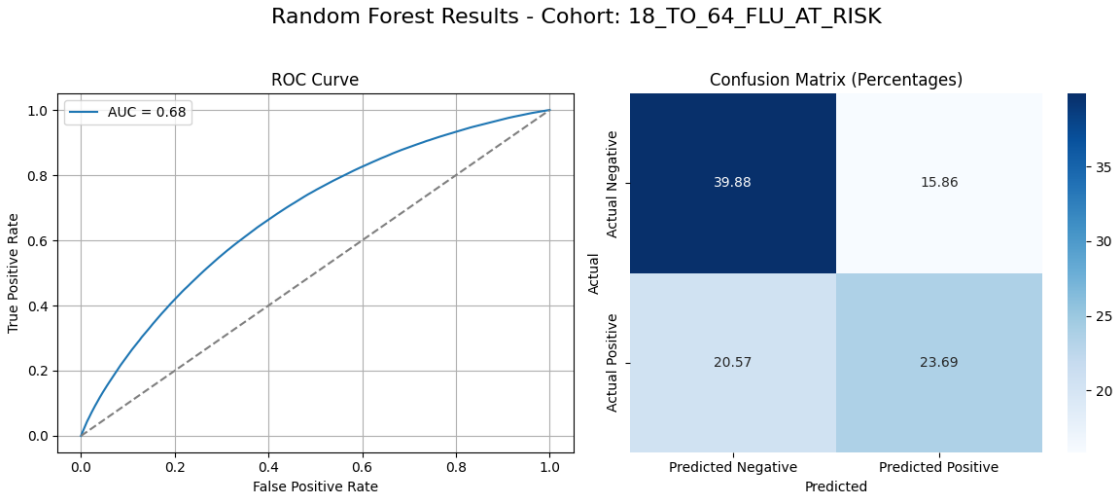
```
AUC Score 0.681424
```




```
Top Feature Importances:
      Feature  Importance
  patient_age      0.8231
     UR8_2022      0.0930
SIMD_quintile      0.0667
  patient_sex      0.0171
/tmp/ipykernel_2887/1701446235.py:91: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(
```

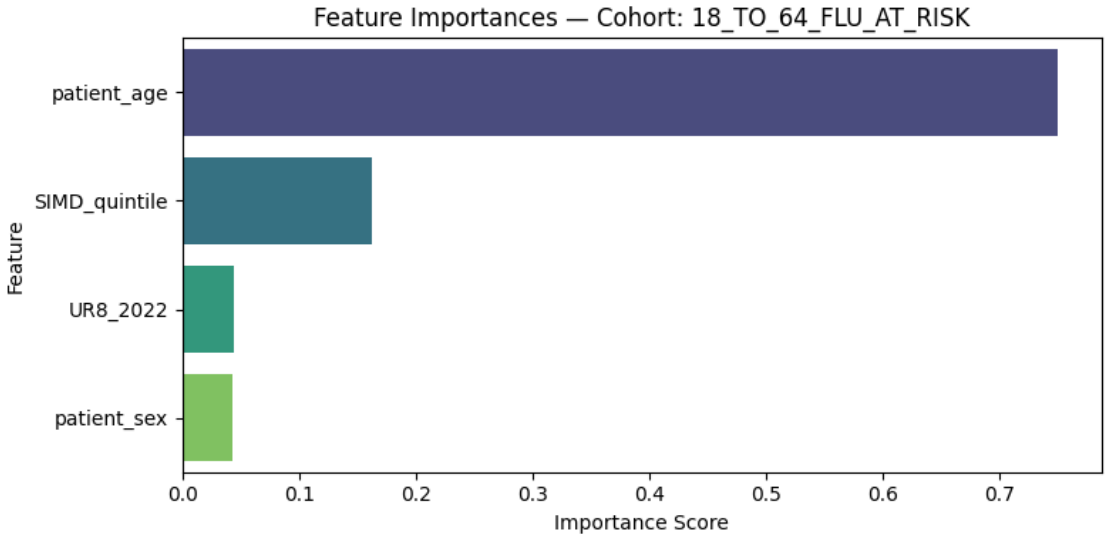



```
======== Cohort: 18_TO_64_FLU_AT_RISK =========
Train size: 1849322
Test size: 792567

Model Performance Metrics:
   Metric    Value
 Accuracy 0.635742
Precision 0.599040
   Recall 0.535248
 F1 Score 0.565350
AUC Score 0.678051
```



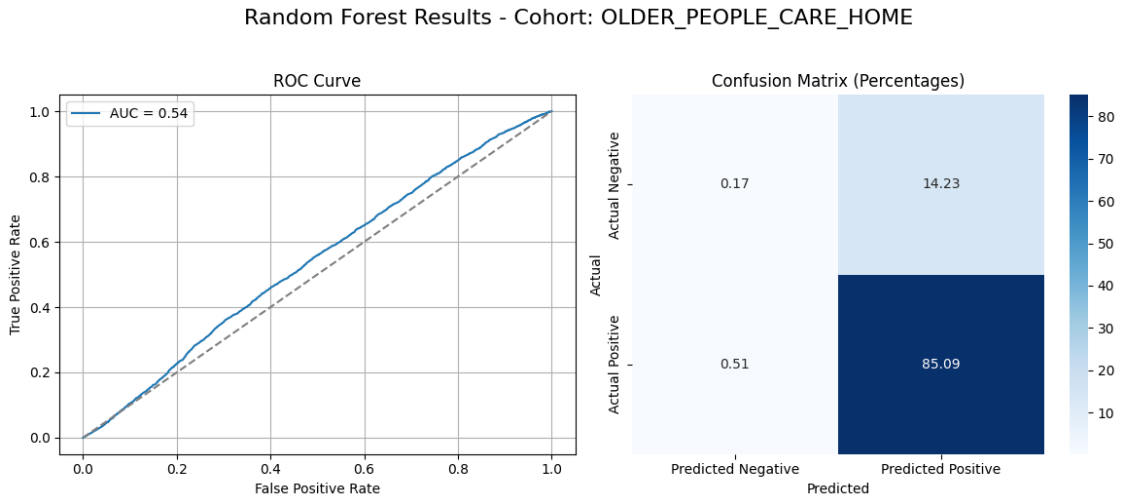
```
Top Feature Importances:
      Feature  Importance
  patient_age      0.7505
SIMD_quintile      0.1624
     UR8_2022      0.0440
  patient_sex      0.0431
```

```
/tmp/ipykernel_2887/1701446235.py:91: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(
```

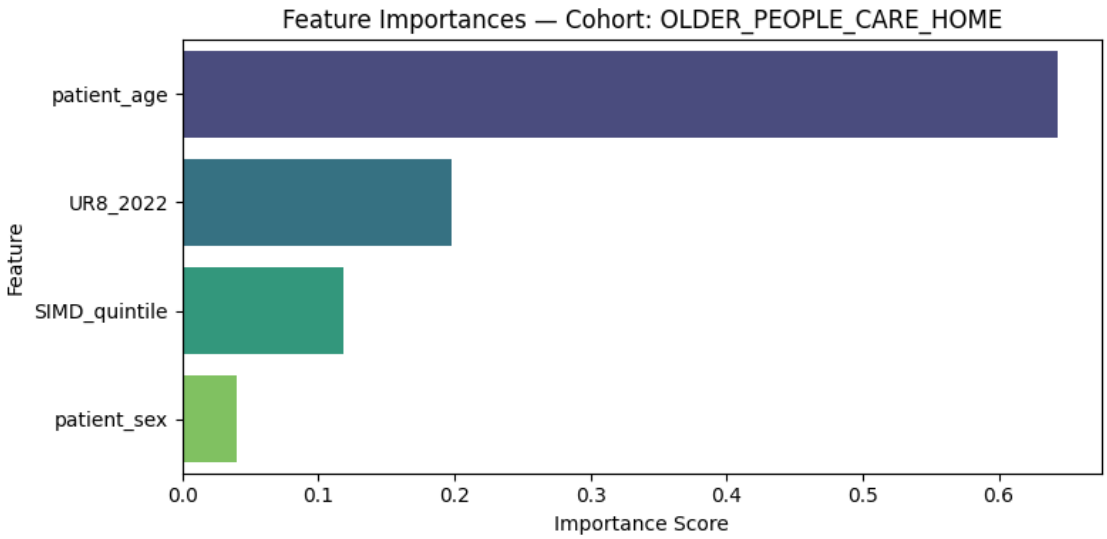



```
======== Cohort: OLDER_PEOPLE_CARE_HOME =========
Train size: 59486
Test size: 25494

Model Performance Metrics:
   Metric    Value
 Accuracy 0.852554
Precision 0.856714
   Recall 0.993997
 F1 Score 0.920264
```

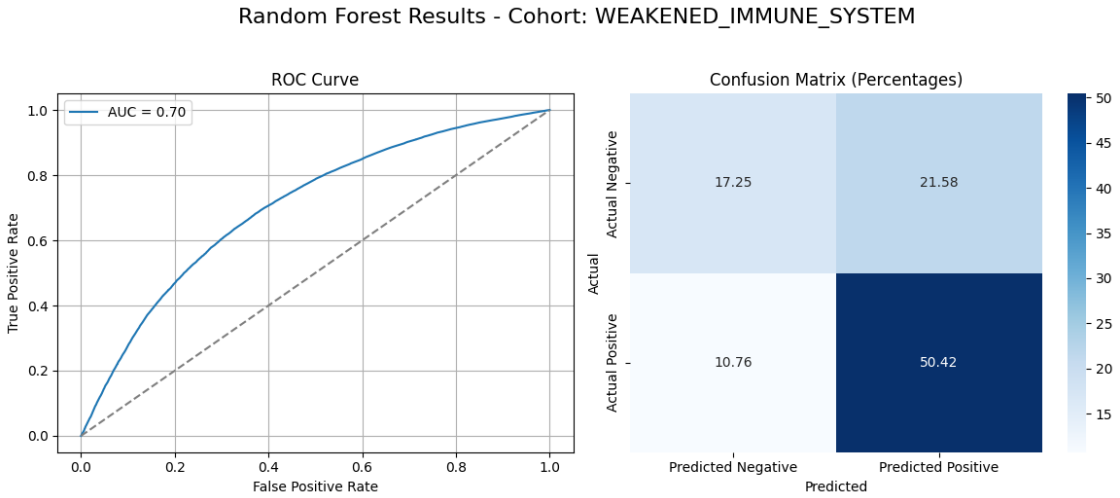
```
AUC Score 0.538418
```




```
Top Feature Importances:
      Feature  Importance
  patient_age      0.6436
     UR8_2022      0.1982
SIMD_quintile      0.1183
  patient_sex      0.0398
/tmp/ipykernel_2887/1701446235.py:91: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(
```




```
======== Cohort: WEAKENED_IMMUNE_SYSTEM =========
Train size: 230736
Test size: 98887

Model Performance Metrics:
   Metric    Value
 Accuracy 0.676631
Precision 0.700299
   Recall 0.824104
 F1 Score 0.757174
AUC Score 0.704023
```



```
Top Feature Importances:
      Feature  Importance
  patient_age      0.7956
SIMD_quintile      0.1063
     UR8_2022      0.0779
  patient_sex      0.0202
/tmp/ipykernel_2887/1701446235.py:91: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(
```