X1136010 黃偉祥 HW3

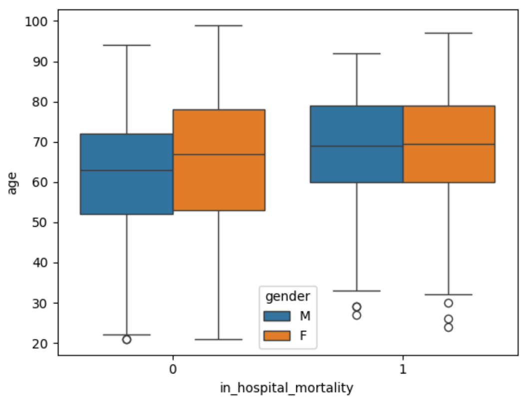
Data Preprocessing Description

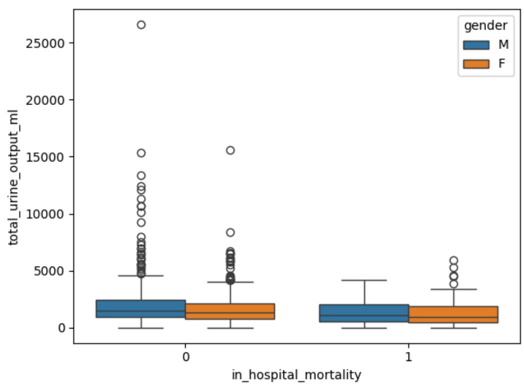
Encoding(Same as HW2)

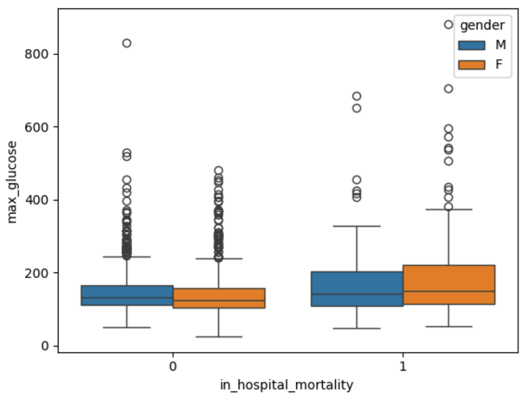
- Encode the Male and female into 0 and 1
- Encode race into range 1~5

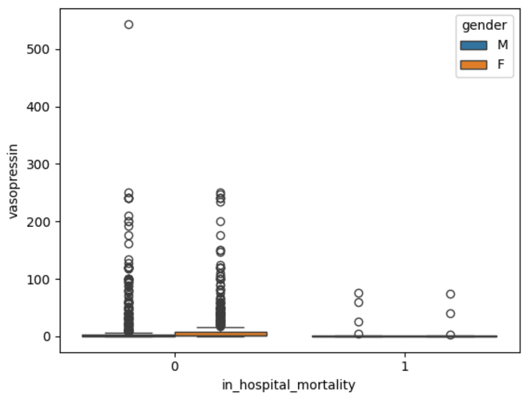
Outliers(Same as HW2)

- age > 90, this is special case, because too old may die anyway regardless the diseases
- total_urine_output_ml > 5000, too much urine output within 30 hours is not really possible
- max_glucose > 600, use box plot to determine
- vasopressin > 100, use box plot to determine

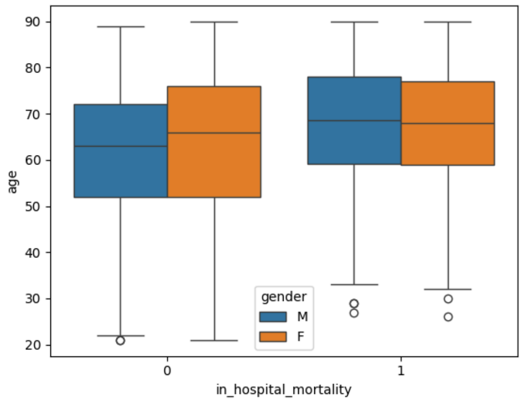


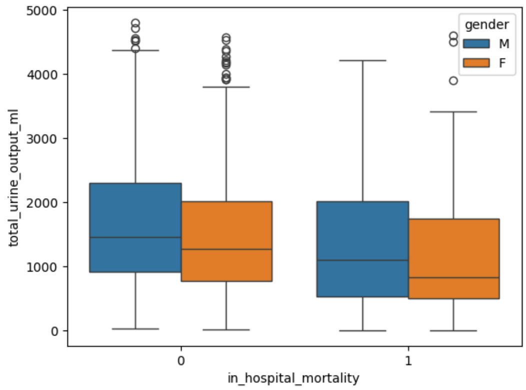


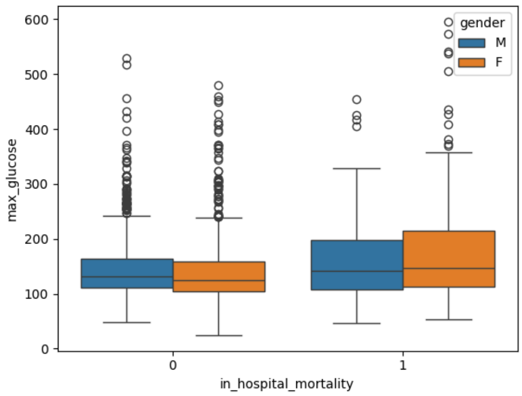


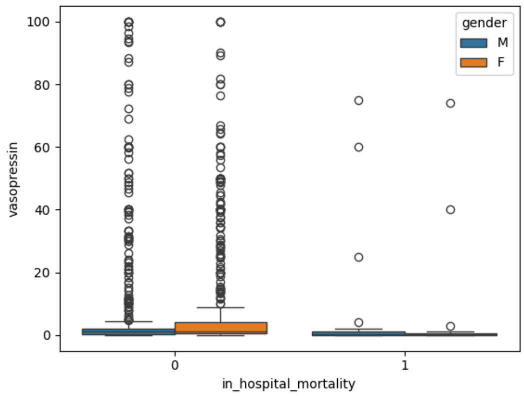


After delete outliers









Apply Z-score(Same as HW2)

```
outlier_mask = (z.abs() >= 3).any(axis=1)
df_outliers   = df[outlier_mask]
df_clean      = df[~outlier_mask]
```

- Z-score ≥ 3 does not detect any outliers

Fill in missing (Same as HW2)

Use **K-Nearest Neighbors** to fill in missing value based on different classes

- Because the classes are very imbalanced, so we need to fill in missing value based on different classes or it will make class 1 missing value filled in with class 0 features
- First separate the dataset into 2 dataset (class_0, class_1) then fill in missing value using K-Nearest Neighbors
- Then concat both and done

Sampling for data imbalance

- The class of the data are very imbalance
  - 1400 for class 0, 275 for class 1 (after data cleaning and fill in missing)

- I use SMOTE to do over sampling to make both have same data amounts

Standardize

- To avoids bias due to units
  - Without scaling, variables measured in different units (e.g., mmHg vs. years) could disproportionately influence the model.
- To Improves model performance
  - Features with large ranges can dominate distance-based or gradient-based calculations if not standardised.

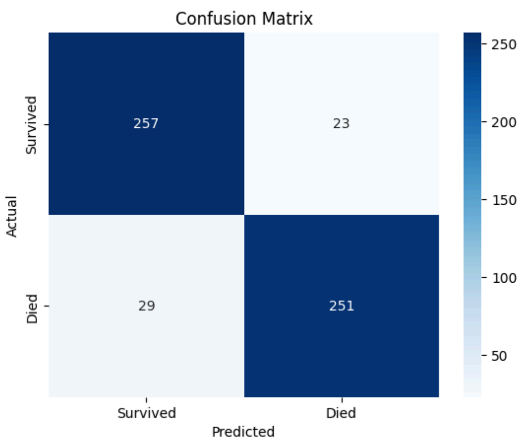
Model architecture

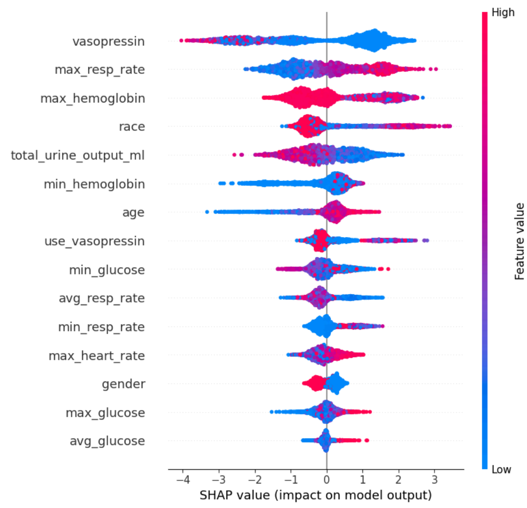
- Split Data set into ratio `8/2`, `80%` for training set and `20%` for testing/validation set
- Compare basic 3 models and select the best model, `cross validation = 10`

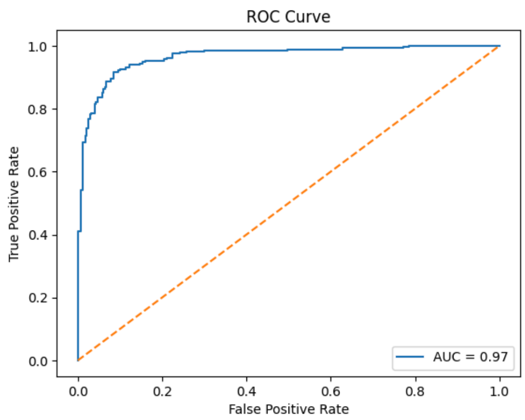
```
Random Forest AUC: 0.975 (±0.009)
XGBoost AUC: 0.978 (±0.007)
Logistic Regression AUC: 0.832 (±0.037)
```

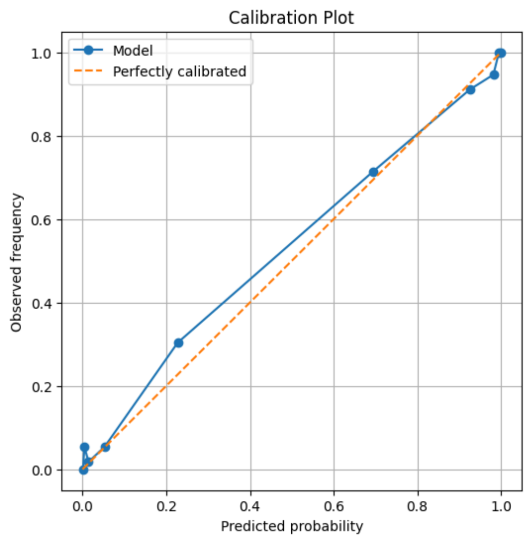
- Choose XGBoost because it has the highest AUC value
  - 200 number of estimators, max depth = 5, learning rate = 0.1

```
Evaluation metrics:

Accuracy           0.907143
Sensitivity (Recall)  0.896429
Specificity         0.917857
Precision           0.916058
F1-Score            0.906137
AUC-ROC             0.965689

Importances(>0.1):
vasopressin       : 0.215882
use_vasopressin    : 0.118857
max_hemoglobin    : 0.113150
race          : 0.109770
```









- Then use `RandomizedSearchCV` to search for better parameters

```
from sklearn.model_selection import RandomizedSearchCV
param_dist = {
    'eval_metric' : ['auc'],
    'n_estimators' : stats.randint(50,500),
    'learning_rate': stats.uniform(0.01, 0.3),
    'max_depth': stats.randint(3,13),
    'min_child_weight': stats.randint(1, 11),
    'subsample': stats.uniform(0.5, 0.5),
    'colsample_bytree': stats.uniform(0.5, 0.5),
    'gamma': stats.uniform(0, 0.5),
    'scale_pos_weight': [sum(y==0)/sum(y==1)]
}
model = XGBClassifier(random_state=42)
search = RandomizedSearchCV(
```

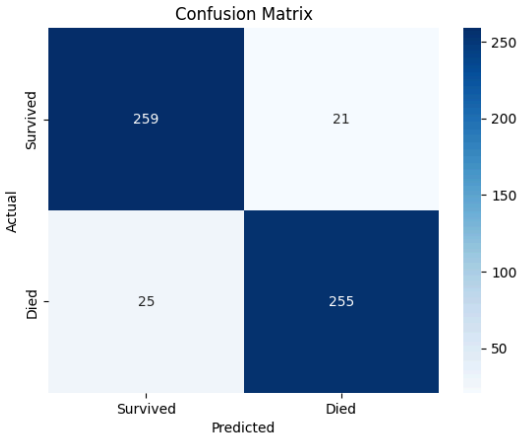
```
    estimator=model,
    param_distributions=param_dist,
    n_iter=5000,
    scoring='roc_auc',
    cv=5,
    verbose=2,
    n_jobs=-1
)
search.fit(X_train, y_train)
```

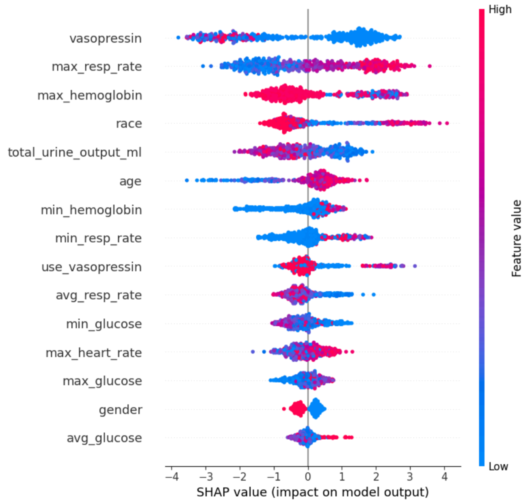
- Best parameters found

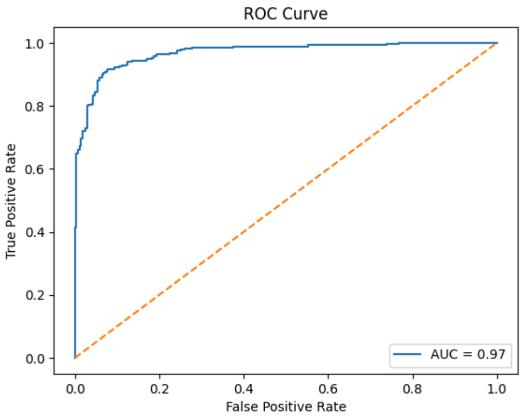
```
Best params: {
'colsample_bytree': np.float64(0.7204789960803597),
'eval_metric': 'auc',
'gamma': np.float64(0.0028207188158109187),
'learning_rate': np.float64(0.09624593233299222),
'max_depth': 10,
'min_child_weight': 1,
'n_estimators': 494,
'scale_pos_weight': 1.0,
'subsample': np.float64(0.8661020114176173)}
```

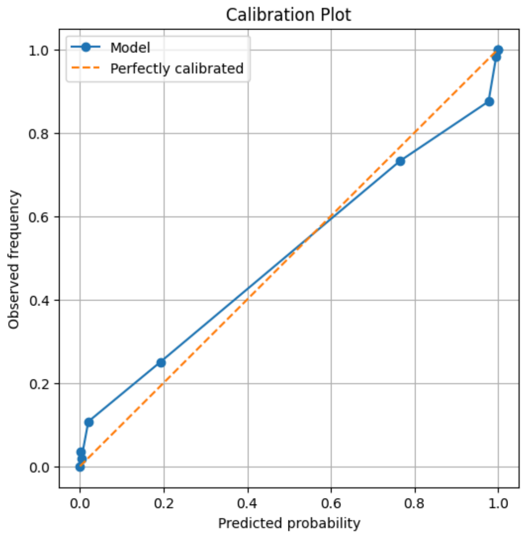
- Result & Evaluations

```
Accuracy            0.917857
Sensitivity (Recall)  0.910714
Specificity          0.925000
Precision            0.923913
F1-Score             0.917266
AUC-ROC              0.968750

Importances(>0.1): (only show greater than 0.1)
vasopressin      : 0.202537
max_hemoglobin : 0.119903
race         : 0.116120
```









- From the SHAP graph:
  - Low vasopressin, low urine output ,low min hemoglobin, high max resp rate, high age may lead to class 1
- Result on many evaluation metrics have increased
  - False positive and False Negative have decrease

Fairness analysis

- Female/Male ratio: 1236/1004

```
Male group performance:
                Value
Accuracy         0.924901
Sensitivity (Recall)  0.907407
```

```
Specificity         0.937931
Precision           0.915888
F1-Score            0.911628
AUC-ROC             0.969157

Female group performance:
                     Value
Accuracy            0.912052
Sensitivity (Recall)  0.912791
Specificity         0.911111
Precision           0.928994
F1-Score            0.920821
AUC-ROC             0.967916
```

- The model have similar performance on both gender group with **above** evaluation metrics, while have some slightly difference at **below** metrics
- Male group:
  - Detection Prevalence (DP): `0.4229`
    Equality of Opportunity: (Recall)
    `0.9074`
    Equiality of Odds:
    FNR:
    `0.0926`
    FPR:
    `0.0621`
- Female group:
  - Detection Prevalence (DP): `0.5505`
    Equality of Opportunity: (Recall)
    `0.9128`
    Equiality of Odds:
    FNR:
    `0.0872`
    FPR:
    `0.0889`
- DP for both gender group is close to 50%, but female is higher
  - It may cause by the Female have more samples than Male
  - Since both are close to 50%, I think is acceptable but also exist the risk that model think the Female is more weighted than Male
- FPR of Female is higher than Male
  - More female been incorrect classified
  - May cause to Female have more samples also
- Recall and FNR of both gender group are very similar
  - Model are fair on both gender group on this part

Ablation study

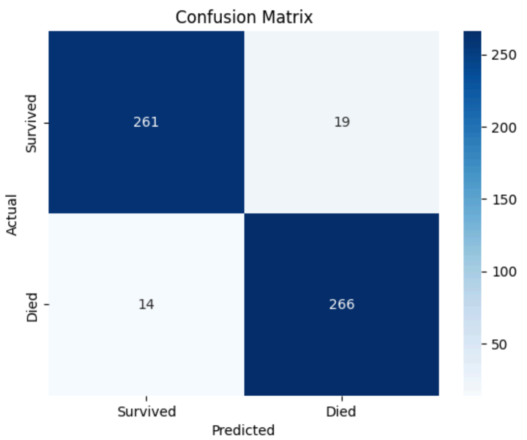
- Duel with imbalance data can increase model performance
    - Because the original dataset is very imbalance, if we use the original dataset will only get a result as follow:
        - Accuracy 0.865672
          Sensitivity (Recall) 0.509091
          Specificity 0.935714
          Precision 0.608696
          F1-Score 0.554455
          AUC-ROC 0.884610
        - Because the samples of class 1 is very less, and class 1 will make very inaccurate predictions
- Delete useless(unimportant) features can let model have better results
    - Some feature is useless and not importance and may decrease model performance
    - Too much features may required larger model size

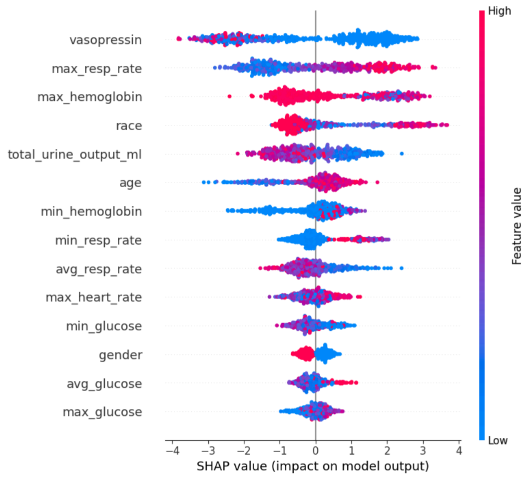
```
Original AUC = 0.9836862244897959
Drop age, AUC-ROC: 0.9803188775510203
Drop gender, AUC-ROC: 0.985484693877551
Drop total_urine_output_ml, AUC-ROC: 0.981109693877551
Drop min_hemoglobin, AUC-ROC: 0.9831505102040816
Drop max_hemoglobin, AUC-ROC: 0.9804591836734694
Drop max_heart_rate, AUC-ROC: 0.9837882653061225
Drop min_resp_rate, AUC-ROC: 0.9834438775510205
Drop avg_resp_rate, AUC-ROC: 0.981938775510204
Drop max_resp_rate, AUC-ROC: 0.9774617346938775
Drop vasopressin, AUC-ROC: 0.9796938775510206
Drop race, AUC-ROC: 0.9809948979591837
Drop min_glucose, AUC-ROC: 0.9846683673469387
Drop max_glucose, AUC-ROC: 0.983966836734694
Drop avg_glucose, AUC-ROC: 0.983609693877551
Drop use_vasopressin, AUC-ROC: 0.980204081632653

Decrease:
 ['age', 'gender', 'total_urine_output_ml', 'min_hemoglobin',
 'max_hemoglobin', 'max_heart_rate', 'max_resp_rate', 'vasopressin',
 'race', 'min_glucose', 'avg_glucose', 'use_vasopressin']

Increase:
 ['min_resp_rate', 'avg_resp_rate', 'max_glucose']
```

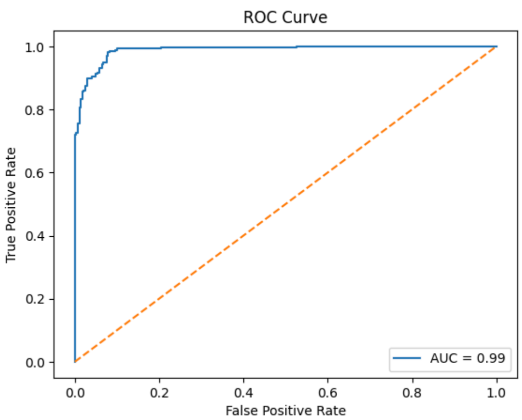
    - Check every features that the AUC result before and after dropping the feature and delete the features that will increase model performance if dropped
        - In this case, we will delete `min_resp_rate` , `avg_resp_rate` , `max_glucose` because after dropping these 3 features will increase model performance!

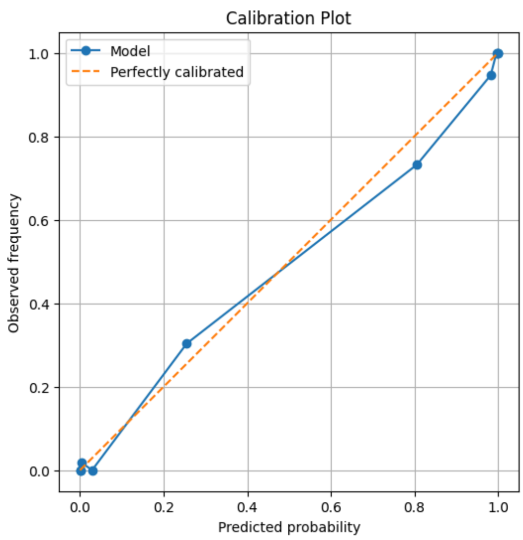
	Value
Accuracy	0.941071
Sensitivity (Recall)	0.950000
Specificity	0.932143
Precision	0.933333
F1-Score	0.941593
AUC-ROC	0.988367

- Performance increased after dropping the useless features!!!









Findings

- When the dataset is very imbalance, we need to do oversampling to let the model learn both classes
  - If one of the class have very less sample, it may let the model just predict as another class and also can have better accuracy. (If we only checking accuracy)

- Use more evaluation metrics
    - If we using only accuracy to check the model performance, then may have model bias due to the model may learn "shortcut" when the dataset is not balance, have bias, ...
- Drop the useless features
    - We can increase the model performance after we found and delete the **redundancy** features
- Random search
    - We can tune our model by using random search cv to get a better model's parameters, and it may increase in the model performances