Oversampling using SMOTE

In [26]:
```
#from collections import Counter
#from imblearn.over_sampling import SMOTE
```

In [27]:
```
#counter = Counter(y_train)
#print(f"before oversampling: {counter}")
#smt = SMOTE()
#X_train , y_train = smt.fit_resample(X_train , y_train)
#counter = Counter(y_train)
#print(f"after oversampling : {counter}")
```

Feature scaling

In [28]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train_final = sc.fit_transform(X_train)
X_test_final = sc.transform(X_test)
```

In [29]:
```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
```

In [30]:
```
# Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier
rf_classifier = RandomForestClassifier(n_estimators = 20, criterion = 'entropy', class_weight = "balanced_subsample",random_state = 51)
rf_classifier.fit(X_train_final, y_train)
y_pred = rf_classifier.predict(X_test_final)
accuracy_score(y_test, y_pred)
```

Out[30]: 0.635

In [31]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support
```

```
from sklearn.ensemble import RandomForestClassifier
rf_classifier = RandomForestClassifier(n_estimators = 20, criterion = 'entropy', class_weight = "balanced_subsample",random_state = 51)
rf_classifier.fit(X_train_final, y_train)
y_pred = rf_classifier.predict(X_test_final)
accuracy_score(y_test, y_pred)
```

Out[30]: 0.635

In [31]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.66      0.86      0.75       497
           1       0.54      0.26      0.35       303

    accuracy                           0.64       800
   macro avg       0.60      0.56      0.55       800
weighted avg       0.61      0.64      0.60       800
```

In [32]:
```
# XGBoost Classifier
from xgboost import XGBClassifier
xgb_classifier = XGBClassifier(random_state=0)
xgb_classifier.fit(X_train_final, y_train)
y_pred_xgb = xgb_classifier.predict(X_test_final)
accuracy_score(y_test, y_pred_xgb)
```

Out[32]: 0.62125

In [33]:
```
print(classification_report(y_test, y_pred_xgb))
```

```
              precision    recall  f1-score   support

           0       0.64      0.90      0.75       497
           1       0.50      0.17      0.25       303

    accuracy                           0.62       800
   macro avg       0.57      0.53      0.50       800
weighted avg       0.59      0.62      0.56       800
```

Support vector Machine

In [34]:
```
# Support vector classifier
from sklearn.svm import SVC
svc_classifier = SVC(class_weight = "balanced" )
svc_classifier.fit(X_train_final, y_train)
y_pred_scv = svc_classifier.predict(X_test_final)
accuracy_score(y_test, y_pred_scv)
```

Out[34]: 0.6225

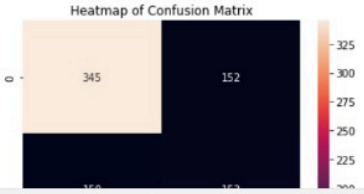
In [35]:
```
print(classification_report(y_test, y_pred_scv))
```

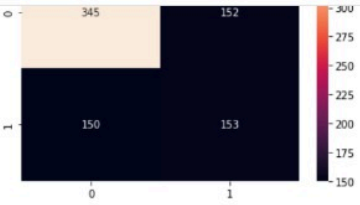
```
              precision    recall  f1-score   support

           0       0.70      0.69      0.70       497
           1       0.50      0.50      0.50       303

    accuracy                           0.62       800
   macro avg       0.60      0.60      0.60       800
weighted avg       0.62      0.62      0.62       800
```

In [36]:
```
cm = confusion_matrix(y_test, y_pred_scv)
plt.title('Heatmap of Confusion Matrix', fontsize = 12)
sns.heatmap(cm, annot = True, fmt = "d")
plt.show()
```





Hyperparameter Tuning with Support vector Machine

In [37]:
```
# defining parameter range
param_grid = {'C': [0.1, 1, 10, 100, 200 , 400 , 600 , 800],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf']}
```

In [38]:
```
from sklearn.model_selection import GridSearchCV
```

In [39]:
```
grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3)

# fitting the model for grid search
grid.fit(X_train_final, y_train)
```

```
Fitting 5 folds for each of 40 candidates, totalling 200 fits
[CV 1/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.628 total time=   0.2s
[CV 2/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.630 total time=   0.2s
[CV 3/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.630 total time=   0.2s
[CV 4/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.630 total time=   0.2s
[CV 5/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.627 total time=   0.2s
[CV 1/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.628 total time=   0.1s
[CV 2/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.630 total time=   0.1s
[CV 3/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.630 total time=   0.2s
[CV 4/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.630 total time=   0.1s
[CV 5/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.627 total time=   0.1s
[CV 1/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.628 total time=   0.1s
[CV 2/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.630 total time=   0.1s
```

```
[CV 2/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.630 total time=   0.1s
[CV 3/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.630 total time=   0.1s
[CV 4/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.630 total time=   0.1s
[CV 5/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.627 total time=   0.1s
[CV 1/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.628 total time=   0.6s
[CV 2/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.630 total time=   0.4s
[CV 3/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.630 total time=   0.1s
[CV 4/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.630 total time=   0.1s
[CV 5/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.627 total time=   0.1s
[CV 1/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.628 total time=   0.1s
[CV 2/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.630 total time=   0.1s
[CV 3/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.630 total time=   0.1s
[CV 4/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.630 total time=   0.1s
[CV 5/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.627 total time=   0.1s
[CV 1/5] END .........C=1, gamma=1, kernel=rbf;, score=0.647 total time=   0.2s
[CV 2/5] END .........C=1, gamma=1, kernel=rbf;, score=0.625 total time=   0.2s
[CV 3/5] END .........C=1, gamma=1, kernel=rbf;, score=0.627 total time=   0.2s
[CV 4/5] END .........C=1, gamma=1, kernel=rbf;, score=0.609 total time=   0.2s
[CV 5/5] END .........C=1, gamma=1, kernel=rbf;, score=0.622 total time=   0.2s
[CV 1/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.644 total time=   0.1s
[CV 2/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.649 total time=   0.1s
[CV 3/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.676 total time=   0.1s
[CV 4/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.651 total time=   0.1s
[CV 5/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.668 total time=   0.1s
[CV 1/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.628 total time=   0.1s
[CV 2/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.630 total time=   0.1s
[CV 3/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.630 total time=   0.1s
[CV 4/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.630 total time=   0.2s
[CV 5/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.627 total time=   0.1s
[CV 1/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.628 total time=   0.1s
[CV 2/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.630 total time=   0.1s
[CV 3/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.630 total time=   0.1s
[CV 4/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.630 total time=   0.1s
[CV 5/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.627 total time=   0.1s
[CV 1/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.628 total time=   0.1s
[CV 2/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.630 total time=   0.1s
[CV 3/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.630 total time=   0.1s
[CV 4/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.630 total time=   0.1s
[CV 5/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.627 total time=   0.1s
[CV 1/5] END .........C=10, gamma=1, kernel=rbf;, score=0.610 total time=   0.2s
[CV 2/5] END .........C=10, gamma=1, kernel=rbf;, score=0.609 total time=   0.2s
[CV 3/5] END .........C=10, gamma=1, kernel=rbf;, score=0.609 total time=   0.2s
[CV 4/5] END .........C=10, gamma=1, kernel=rbf;, score=0.609 total time=   0.2s
[CV 5/5] END .........C=10, gamma=1, kernel=rbf;, score=0.627 total time=   0.2s
```

Out[39]:
```
GridSearchCV(estimator=SVC(),
             param_grid={'C': [0.1, 1, 10, 100, 200, 400, 600, 800],
                         'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
                         'kernel': ['rbf']},
             verbose=3)
```

In [40]:
```
# print best parameter after tuning
print(grid.best_params_)

# print how our model looks after hyper-parameter tuning
print(grid.best_estimator_)
```

```
{'C': 100, 'gamma': 0.01, 'kernel': 'rbf'}
SVC(C=100, gamma=0.01)
```

In [41]:
```
# Support vector classifier
from sklearn.svm import SVC
svc_classifier = SVC(class_weight = "balanced"  , C=100, gamma=0.01)
svc_classifier.fit(X_train_final, y_train)
y_pred_scv = svc_classifier.predict(X_test_final)
accuracy_score(y_test, y_pred_scv)
```

Out[41]: 0.6325

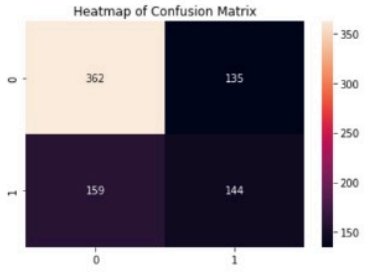
In [42]:
```
print(classification_report(y_test, y_pred_xgb))
```

```
              precision    recall  f1-score   support

           0       0.64      0.90      0.75       497
           1       0.50      0.17      0.25       303

    accuracy                           0.62       800
   macro avg       0.57      0.53      0.50       800
weighted avg       0.59      0.62      0.56       800
```

In [43]:
```
cm = confusion_matrix(y_test, y_pred_scv)
plt.title('Heatmap of Confusion Matrix', fontsize = 12)
sns.heatmap(cm, annot = True , fmt = "d")
plt.show()
```



In [44]:

```
## Pickle
from sklearn.svm import SVC
import pickle

# save model
pickle.dump(svc_classifier, open('model.pkl', 'wb'))

# load model
water_quality_model = pickle.load(open('model.pkl', 'rb'))

# predict the output
y_pred =water_quality_model.predict(X_test_final)

# confusion matrix
print('Confusion matrix of Support vector Machine : \n',confusion_matrix(y_test, y_pred),'\n')
```

```
Confusion matrix of Support vector Machine :
 [[362 135]
 [159 144]]
```