In [65]:
```
import numpy as np
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.metrics import classification_report
import warnings
warnings.filterwarnings('ignore')
```

In [66]:
```
x,y=make_classification(n_samples=1000,n_features=10,n_informative=2,n_redundant
np.unique(y,return_counts=True)
```

Out[66]: (array([0, 1]), array([900, 100]))

In [67]:
```
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.3,stratify=y,rand
```

In [68]:
```
print(X_train.shape)
print(y_train.shape)
```

```
(1226, 10)
(700,)
```

In [69]:
```
print(x_test.shape)
print(y_test.shape)
```

```
(300, 10)
(300,)
```

Experiment 1: Train Logistic Regression Classifier

In [70]:
```
log_reg=LogisticRegression(C=1, solver='liblinear')
log_reg.fit(x_train,y_train)
y_pred_log_reg=log_reg.predict(x_test)
print(classification_report(y_test,y_pred_log_reg))
```

```
              precision    recall  f1-score   support

           0       0.95      0.96      0.95       270
           1       0.60      0.50      0.55        30

    accuracy                           0.92       300
   macro avg       0.77      0.73      0.75       300
weighted avg       0.91      0.92      0.91       300
```

In [71]:
```
rf_clf=RandomForestClassifier(n_estimators=30,max_depth=3)
rf_clf.fit(x_train,y_train)
y_pred_rf=rf_clf.predict(x_test)
print(classification_report(y_test,y_pred_rf))
```

```
                precision    recall  f1-score   support

           0       0.96      1.00      0.98       270
           1       0.95      0.67      0.78        30

    accuracy                           0.96       300
   macro avg       0.96      0.83      0.88       300
weighted avg       0.96      0.96      0.96       300
```

Experiment 3: Train XGBoost

In [72]:
```
xgb_clf=XGBClassifier(use_label_encoder=False, eval_metric='logloss')
xgb_clf.fit(x_train,y_train)
y_pred_xgb=xgb_clf.predict(x_test)
print(classification_report(y_test,y_pred_xgb))
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       270
           1       0.96      0.80      0.87        30

    accuracy                           0.98       300
   macro avg       0.97      0.90      0.93       300
weighted avg       0.98      0.98      0.98       300
```

Experiment 4: Handle class imbalance using SMOTETomek and then Train XGBoost

In [73]:
```
from imblearn.combine import SMOTETomek

smt=SMOTETomek(random_state=0)
x_train_res,y_train_res=smt.fit_resample(x_train,y_train)

np.unique(y_train_res,return_counts=True)
```

Out[73]:
```
(array([0, 1]), array([613, 613]))
```

In [74]:
```
xgb_clf=XGBClassifier(use_label_encoder=False, eval_metric='logloss')
xgb_clf.fit(x_train_res,y_train_res)
y_pred_xgb=xgb_clf.predict(x_test)
print(classification_report(y_test,y_pred_xgb))
```

```
              precision    recall  f1-score   support

           0       0.98      0.97      0.98       270
           1       0.76      0.83      0.79        30

    accuracy                           0.96       300
   macro avg       0.87      0.90      0.88       300
weighted avg       0.96      0.96      0.96       300
```

Track Experiments Using MLFlow

In [75]:
```
models=[
    (
        "Logistic Regression",
        LogisticRegression(C=1, solver='liblinear'),
        (x_train,y_train),
        (x_test,y_test)
    ),
    (
        "Random Forest",
        RandomForestClassifier(n_estimators=30,max_depth=3),
        (x_train,y_train),
        (x_test,y_test)
    ),
    (
        "XGBClassifier",
        XGBClassifier(use_label_encoder=False,eval_metric='logloss'),
        (x_train,y_train),
        (x_test,y_test)
    ),
    (
        "XGBClassifier With SMOTE",
        XGBClassifier(use_label_encoder=False,eval_metric='logloss'),
        (x_train_res,y_train_res),
        (x_test,y_test)
    )
]
```

In [76]:
```
reports=[]

for model_name, model,train_set,test_set in models:
    x_train=train_set[0]
    y_trian=train_set[1]
    x_test=test_set[0]
    y_test=test_set[1]

    model.fit(x_train,y_train)
    y_pred=model.predict(x_test)
    report=classification_report(y_test,y_pred,output_dict=True)
    reports.append(report)
```

```
---------------------------------------------------------------------------
XGBoostError                              Traceback (most recent call last)
Cell In[76], line 9
      6 x_test=test_set[0]
      7 y_test=test_set[1]
----> 9 model.fit(x_train,y_train)
     10 y_pred=model.predict(x_test)
     11 report=classification_report(y_test,y_pred,output_dict=True)

File D:\New folder\Lib\site-packages\xgboost\core.py:729, in require_keyword_arg
s.<locals>.throw_if.<locals>.inner_f(*args, **kwargs)
    727 for k, arg in zip(sig.parameters, args):
    728     kwargs[k] = arg
--> 729 return func(**kwargs)

File D:\New folder\Lib\site-packages\xgboost\sklearn.py:1664, in XGBClassifier.fi
t(self, X, y, sample_weight, base_margin, eval_set, verbose, xgb_model, sample_we
ight_eval_set, base_margin_eval_set, feature_weights)
   1659     params["num_class"] = self.n_classes_
   1661 model, metric, params, feature_weights = self._configure_fit(
   1662     xgb_model, params, feature_weights
   1663 )
-> 1664 train_dmatrix, evals = _wrap_evaluation_matrices(
   1665     missing=self.missing,
   1666     X=X,
   1667     y=y,
   1668     group=None,
   1669     qid=None,
   1670     sample_weight=sample_weight,
   1671     base_margin=base_margin,
   1672     feature_weights=feature_weights,
   1673     eval_set=eval_set,
   1674     sample_weight_eval_set=sample_weight_eval_set,
   1675     base_margin_eval_set=base_margin_eval_set,
   1676     eval_group=None,
   1677     eval_qid=None,
   1678     create_dmatrix=self._create_dmatrix,
   1679     enable_categorical=self.enable_categorical,
   1680     feature_types=self.feature_types,
   1681 )
   1683 self._Booster = train(
   1684     params,
   1685     train_dmatrix,
   (...)
   1694     callbacks=self.callbacks,
   1695 )
   1697 if not callable(self.objective):

File D:\New folder\Lib\site-packages\xgboost\sklearn.py:628, in _wrap_evaluation_
matrices(missing, X, y, group, qid, sample_weight, base_margin, feature_weights,
eval_set, sample_weight_eval_set, base_margin_eval_set, eval_group, eval_qid, cre
ate_dmatrix, enable_categorical, feature_types)
    607 def _wrap_evaluation_matrices(
    608     *,
    609     missing: float,
   (...)
    624     feature_types: Optional[FeatureTypes],
    625 ) -> Tuple[Any, List[Tuple[Any, str]]]:
    626     """Convert array_like evaluation matrices into DMatrix.  Perform vali
dation on the
```

```
    627     way."""
--> 628     train_dmatrix = create_dmatrix(
    629         data=X,
    630         label=y,
    631         group=group,
    632         qid=qid,
    633         weight=sample_weight,
    634         base_margin=base_margin,
    635         feature_weights=feature_weights,
    636         missing=missing,
    637         enable_categorical=enable_categorical,
    638         feature_types=feature_types,
    639         ref=None,
    640     )
    642     n_validation = 0 if eval_set is None else len(eval_set)
    644     def validate_or_none(meta: Optional[Sequence], name: str) -> Sequenc
e:

File D:\New folder\Lib\site-packages\xgboost\sklearn.py:1137, in XGBModel._create
_dmatrix(self, ref, **kwargs)
   1135 if _can_use_qdm(self.tree_method, self.device) and self.booster != "gblin
ear":
   1136     try:
-> 1137         return QuantileDMatrix(
   1138             **kwargs, ref=ref, nthread=self.n_jobs, max_bin=self.max_bin
   1139         )
   1140     except TypeError:  # `QuantileDMatrix` supports lesser types than DMa
trix
   1141         pass

File D:\New folder\Lib\site-packages\xgboost\core.py:729, in require_keyword_arg
s.<locals>.throw_if.<locals>.inner_f(*args, **kwargs)
    727 for k, arg in zip(sig.parameters, args):
    728     kwargs[k] = arg
--> 729 return func(**kwargs)

File D:\New folder\Lib\site-packages\xgboost\core.py:1614, in QuantileDMatrix.__i
nit__(self, data, label, weight, base_margin, missing, silent, feature_names, fea
ture_types, nthread, max_bin, ref, group, qid, label_lower_bound, label_upper_bou
nd, feature_weights, enable_categorical, max_quantile_batches, data_split_mode)
   1594     if any(
   1595         info is not None
   1596         for info in (
   (...)
   1607         )
   1608     ):
   1609         raise ValueError(
   1610             "If data iterator is used as input, data like label should be
"
   1611             "specified as batch argument."
   1612         )
-> 1614 self._init(
   1615     data,
   1616     ref=ref,
   1617     label=label,
   1618     weight=weight,
   1619     base_margin=base_margin,
   1620     group=group,
   1621     qid=qid,
   1622     label_lower_bound=label_lower_bound,
```

```
   1623     label_upper_bound=label_upper_bound,
   1624     feature_weights=feature_weights,
   1625     feature_names=feature_names,
   1626     feature_types=feature_types,
   1627     enable_categorical=enable_categorical,
   1628     max_quantile_blocks=max_quantile_batches,
   1629 )

File D:\New folder\Lib\site-packages\xgboost\core.py:1678, in QuantileDMatrix._in
it(self, data, ref, enable_categorical, max_quantile_blocks, **meta)
   1663 config = make_jcargs(
   1664     nthread=self.nthread,
   1665     missing=self.missing,
   1666     max_bin=self.max_bin,
   1667     max_quantile_blocks=max_quantile_blocks,
   1668 )
   1669 ret = _LIB.XGQuantileDMatrixCreateFromCallback(
   1670     None,
   1671     it.proxy.handle,
   (...)
   1676     ctypes.byref(handle),
   1677 )
-> 1678 it.reraise()
   1679 # delay check_call to throw intermediate exception first
   1680 _check_call(ret)

File D:\New folder\Lib\site-packages\xgboost\core.py:572, in DataIter.reraise(sel
f)
    570 exc = self._exception
    571 self._exception = None
--> 572 raise exc

File D:\New folder\Lib\site-packages\xgboost\core.py:553, in DataIter._handle_exc
eption(self, fn, dft_ret)
    550     return dft_ret
    552 try:
--> 553     return fn()
    554 except Exception as e:  # pylint: disable=broad-except
    555     # Defer the exception in order to return 0 and stop the iteration.
    556     # Exception inside a ctype callback function has no effect except
    557     # for printing to stderr (doesn't stop the execution).
    558     tb = sys.exc_info()[2]

File D:\New folder\Lib\site-packages\xgboost\core.py:640, in DataIter._next_wrapp
er.<locals>.<lambda>()
    638     self._temporary_data = None
    639 # pylint: disable=not-callable
--> 640 return self._handle_exception(lambda: int(self.next(input_data)), 0)

File D:\New folder\Lib\site-packages\xgboost\data.py:1654, in SingleBatchInternal
Iter.next(self, input_data)
   1652     return False
   1653 self.it += 1
-> 1654 input_data(**self.kwargs)
   1655 return True

File D:\New folder\Lib\site-packages\xgboost\core.py:729, in require_keyword_arg
s.<locals>.throw_if.<locals>.inner_f(*args, **kwargs)
    727 for k, arg in zip(sig.parameters, args):
    728     kwargs[k] = arg
```

```
--> 729 return func(**kwargs)

File D:\New folder\Lib\site-packages\xgboost\core.py:629, in DataIter._next_wrapp
er.<locals>.input_data(data, feature_names, feature_types, **kwargs)
    627 self._temporary_data = (new, cat_codes, feature_names, feature_types)
    628 dispatch_proxy_set_data(self.proxy, new, cat_codes)
--> 629 self.proxy.set_info(
    630     feature_names=feature_names,
    631     feature_types=feature_types,
    632     **kwargs,
    633 )
    634 self._data_ref = ref

File D:\New folder\Lib\site-packages\xgboost\core.py:729, in require_keyword_arg
s.<locals>.throw_if.<locals>.inner_f(*args, **kwargs)
    727 for k, arg in zip(sig.parameters, args):
    728     kwargs[k] = arg
--> 729 return func(**kwargs)

File D:\New folder\Lib\site-packages\xgboost\core.py:961, in DMatrix.set_info(sel
f, label, weight, base_margin, group, qid, label_lower_bound, label_upper_bound,
feature_names, feature_types, feature_weights)
    958 from .data import dispatch_meta_backend
    960 if label is not None:
--> 961     self.set_label(label)
    962 if weight is not None:
    963     self.set_weight(weight)

File D:\New folder\Lib\site-packages\xgboost\core.py:1099, in DMatrix.set_label(s
elf, label)
   1090 """Set label of dmatrix
   1091
   1092 Parameters
   (...)
   1095     The label information to be set into DMatrix
   1096 """
   1097 from .data import dispatch_meta_backend
-> 1099 dispatch_meta_backend(self, label, "label", "float")

File D:\New folder\Lib\site-packages\xgboost\data.py:1586, in dispatch_meta_backe
nd(matrix, data, name, dtype)
   1584     return
   1585 if _is_np_array_like(data):
-> 1586     _meta_from_numpy(data, name, dtype, handle)
   1587     return
   1588 if _is_arrow(data):

File D:\New folder\Lib\site-packages\xgboost\data.py:1533, in _meta_from_numpy(da
ta, field, dtype, handle)
   1531     raise ValueError("Masked array is not supported.")
   1532 interface_str = array_interface(data)
-> 1533 _check_call(_LIB.XGDMatrixSetInfoFromInterface(handle, c_str(field), inte
rface_str))

File D:\New folder\Lib\site-packages\xgboost\core.py:310, in _check_call(ret)
    299 """Check the return value of C API call
    300
    301 This function will raise exception when error occurs.
   (...)
    307     return value from API calls
```

```
    308 """
    309 if ret != 0:
--> 310     raise XGBoostError(py_str(_LIB.XGBGetLastError()))

XGBoostError: [12:38:35] C:\actions-runner\_work\xgboost\xgboost\src\data\data.c
c:542: Check failed: this->labels.Size() % this->num_row_ == 0 (700 vs. 0) : Inco
rrect size for labels: (700,1) v.s. 1226
```

In [77]:
```
import mlflow
```

In [80]:
```
mlflow.set_experiment("imbalanced classification")
mlflow.set_tracking_uri("http://localhost:5000")

for i, element in enumerate(models):
    model_name=element[0]
    model=element[1]
    report=reports[i]

    with mlflow.start_run(run_name=model_name):
        mlflow.log_param("model",model_name)
        mlflow.log_metric('accuracy',report['accuracy'])
        mlflow.log_metric('recall_class_1',report['1']['recall'])
        mlflow.log_metric('recall_class_0',report['0']['recall'])
        mlflow.log_metric('f1_score_macro',report['macro avg']['f1-score'])

        if "XGB" in model_name:
            mlflow.xgboost.log_model(model,"model")
        else:
            mlflow.sklearn.log_model(model,"model")
```

```
2025/09/28 12:51:05 WARNING mlflow.models.model: `artifact_path` is deprecated. P
lease use `name` instead.
2025/09/28 12:51:49 WARNING mlflow.models.model: Model logged without a signature
and input example. Please set `input_example` parameter when logging the model to
auto infer the model signature.
🏃 View run Logistic Regression at: http://localhost:5000/#/experiments/148083648
889951264/runs/5c8b51c16b854ee68e60ccc7847c1487
🧪 View experiment at: http://localhost:5000/#/experiments/148083648889951264
2025/09/28 12:51:50 WARNING mlflow.models.model: `artifact_path` is deprecated. P
lease use `name` instead.
2025/09/28 12:51:59 WARNING mlflow.models.model: Model logged without a signature
and input example. Please set `input_example` parameter when logging the model to
auto infer the model signature.
🏃 View run Random Forest at: http://localhost:5000/#/experiments/148083648889951
264/runs/4c1eb18618a1473f990140e0f8a143e9
🧪 View experiment at: http://localhost:5000/#/experiments/148083648889951264
2025/09/28 12:52:00 WARNING mlflow.models.model: `artifact_path` is deprecated. P
lease use `name` instead.
2025/09/28 12:52:09 WARNING mlflow.models.model: Model logged without a signature
and input example. Please set `input_example` parameter when logging the model to
auto infer the model signature.
🏃 View run XGBClassifier at: http://localhost:5000/#/experiments/148083648889951
264/runs/ce37a9789c7c4525b3df180a2e8c512e
🧪 View experiment at: http://localhost:5000/#/experiments/148083648889951264
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[80], line 7
      5 model_name=element[0]
      6 model=element[1]
----> 7 report=reports[i]
      9 with mlflow.start_run(run_name=model_name):
     10     mlflow.log_param("model",model_name)

IndexError: list index out of range
```

In [ ]: