In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import xgboost as xgb
from scipy import stats
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from sklearn.model_selection import train_test_split, RandomizedSearch
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classifi
from sklearn.utils.class_weight import compute_class_weight
from sklearn.datasets import make_classification
from imblearn.over_sampling import SMOTE
from xgboost import XGBClassifier
```

In [2]:
```
# read in data - this is a CSV file of general features for around 730
data_path = "ALL_full_star_table_all_types.txt"
data = pd.read_csv(data_path, sep='\t', skiprows=6, header=0)
```

```
/opt/apps/intel19/python3/3.9.7/lib/python3.9/site-packages/IPython/c
ore/interactiveshell.py:3444: DtypeWarning: Columns (6,36) have mixed
types.Specify dtype option on import or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

In [3]:
```
labels = data["Type"]
```

In [4]:
```
# replacing -99.99s with NaNs
data = data.replace(-99.99, np.nan)
data = data.replace("-99.99", np.nan)
```

In [5]:
```
high_nan = data.columns[data.isna().mean() > 0.2]

# create a DataFrame with non-NaN columns
df_filtered = data.drop(columns=high_nan)
df_filtered.head()
```

Out[5]:

	ID	Type	Subtype	RA	DECL	ID_OGLE_IV	I	P_1	T0
0	OGLE-BLG-CEP-001	Cep	F	17.570842	-27.398250	BLG611.14.36983	17.395	2.597573	7002.541
1	OGLE-BLG-CEP-002	Cep	F	17.632956	-22.503361	BLG625.32.78667	15.734	2.025573	7000.984
2	OGLE-BLG-CEP-003	Cep	F1	17.745497	-23.723639	BLG632.13.133301	16.424	1.235729	7000.555
3	OGLE-BLG-CEP-004	Cep	12	17.763842	-33.768778	BLG603.29.45415	16.178	0.240046	7000.165
4	OGLE-BLG-CEP-005	Cep	F	17.818625	-23.121861	NaN	15.374	3.795593	7002.172

In [6]:
```
non_numeric_columns = df_filtered.select_dtypes(exclude=['number'])
df_numeric = df_filtered.drop(columns=non_numeric_columns)
df_numeric
```

Out[6]:

	RA	DECL	I	P_1	T0_1	A_1
0	17.570842	-27.398250	17.395	2.597573	7002.54120	0.523
1	17.632956	-22.503361	15.734	2.025573	7000.98498	0.730
2	17.745497	-23.723639	16.424	1.235729	7000.55567	0.046
3	17.763842	-33.768778	16.178	0.240046	7000.16541	0.110
4	17.818625	-23.121861	15.374	3.795593	7002.17287	0.409
...	...	...	...	...	...	...
735937	1.028956	-75.017250	17.464	4.227618	7001.52773	0.299
735938	1.102853	-71.079444	18.708	1.065770	7000.11577	0.288
735939	1.161111	-70.477722	18.147	1.746251	7000.80591	0.444
735940	1.214036	-74.588444	16.307	14.912622	7013.31086	0.631
735941	1.319739	-75.067194	17.891	1.842325	7000.66340	0.699

735942 rows × 6 columns

In [7]:
```
features = ['T0_1', 'A_1', 'I', 'P_1']
data = df_numeric[features]
```

In [8]:
```
# dropping NaNs from dataframe
nan_indices = data[data.isna().any(axis=1)].index
data = data.dropna()
```

In [9]:
```
# removing corresponding indices from labels
labels = [label for i, label in enumerate(labels) if i not in nan_indi
classes = np.unique(labels)
data = data.reset_index(drop=True)
```

In [10]:
```
encoder = LabelEncoder()
labels_encoded = encoder.fit_transform(labels)
labels_encoded
```

Out[10]:
```
array([0, 0, 0, ..., 5, 5, 5])
```

In [11]:
```
# smote instance
smote = SMOTE(sampling_strategy='auto', random_state=21)
X_resampled, y_resampled = smote.fit_resample(data, labels_encoded)
```

In [12]:
```
X_train, X_temp, y_train, y_temp = train_test_split(X_resampled,
                                                    y_resampled,
                                                    test_size=0.3,
                                                    random_state=21)

X_val, X_test, y_val, y_test = train_test_split(X_temp,
                                                y_temp,
                                                test_size=0.5,
                                                random_state=21)
```

In [14]:
```
# hyperparameter tuning for random forest
param_dist = {
    'n_estimators': [100, 200, 300, 400],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 4, 5],
    'min_child_weight': [1, 2, 3],
    'gamma': [0, 0.1, 0.2],
}


# hyperparameter tuning for XGBoost
xgb_test = XGBClassifier()

rand_search = RandomizedSearchCV(estimator=xgb_test,
                                 param_distributions=param_dist,
                                 n_iter=20,
                                 cv=5,
                                 scoring="accuracy",
                                 random_state=21,
                                 n_jobs=-1)

rand_search.fit(X_val, y_val)

best_params = rand_search.best_params_
best_params
```

Out[14]:
```
{'n_estimators': 300,
 'min_child_weight': 2,
 'max_depth': 5,
 'learning_rate': 0.2,
 'gamma': 0}
```

In [15]:
```
xgb = XGBClassifier(n_estimators=300,
                    gamma=0,
                    learning_rate=0.2,
                    max_depth=5,
                    min_child_weight=2,
                    random_state=21)
xgb.fit(X_train, y_train)
```

Out[15]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_r
ounds=None,
              enable_categorical=False, eval_metric=None, feature_
types=None,
              gamma=0, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.2, max
_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
```

In [18]:
```
xgb_predictions = xgb.predict(X_test)
xgb_accuracy = accuracy_score(y_test, xgb_predictions)
xgb_accuracy * 100
```

Out[18]:
```
93.64732009134946
```

In [23]:
```
conf_mat_2 = confusion_matrix(y_test, xgb_predictions)
conf_mat_pct_2 = conf_mat_2.astype("float") / conf_mat_2.sum(axis=1)[:

plt.figure(figsize=(16, 14))
ax = sns.heatmap(conf_mat_pct_2, fmt=".2f", cmap="Blues", cbar=False,
            xticklabels=classes,  # Predicted
            yticklabels=classes)

for i in range(len(classes)):
    for j in range(len(classes)):
        count = conf_mat_2[i, j]
        percent = conf_mat_pct_2[i, j]
        text = f"{count} ({percent:.2f}%)"
        color = 'white' if i == j else 'black'  # White for diagonal,
        ax.text(j + 0.5, i + 0.5, text, ha='center', va='center', colo

plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("XGBoost Confusion Matrix")
plt.show()
```

In [24]:
```
conf_mat_2
```

Out[24]:
```
array([[66480,    10,  1437,     0,   479,  4107,  2606],
       [    3, 74796,     8,     0,    63,     0,     0],
       [  747,    40, 62314,     0,  6000,  4864,   955],
       [    0,     0,     1, 74532,     0,     0,     0],
       [  480,   140,  5722,     0, 68107,     0,   790],
       [  621,     0,  1565,     0,     0, 72382,   213],
       [  942,     0,   345,     0,   549,   610, 72233]])
```

In [25]:
```
conf_mat_pct_2
```

Out[25]:
```
array([[8.84995807e+01, 1.33122113e-02, 1.91296476e+00, 0.00000000e+0
0,
        6.37654921e-01, 5.46732518e+00, 3.46916226e+00],
       [4.00694537e-03, 9.99011620e+01, 1.06851877e-02, 0.00000000e+0
0,
        8.41458528e-02, 0.00000000e+00, 0.00000000e+00],
       [9.97063534e-01, 5.33902830e-02, 8.31740523e+01, 0.00000000e+0
0,
        8.00854245e+00, 6.49225841e+00, 1.27469301e+00],
       [0.00000000e+00, 0.00000000e+00, 1.34168757e-03, 9.99986583e+0
1,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [6.37967012e-01, 1.86073712e-01, 7.60509842e+00, 0.00000000e+0
0,
        9.05208735e+01, 0.00000000e+00, 1.04998737e+00],
       [8.30424841e-01, 0.00000000e+00, 2.09277758e+00, 0.00000000e+0
0,
        0.00000000e+00, 9.67919659e+01, 2.84831709e-01],
       [1.26139879e+00, 0.00000000e+00, 4.61977263e-01, 0.00000000e+0
0,
        7.35146427e-01, 8.16829363e-01, 9.67246482e+01]])
```

In [28]:
```
import pickle
```

In [29]:
```
with open('conf_mat_2_XGBoost.pkl','wb') as f:
    pickle.dump(conf_mat_2, f)
```

In [30]:
```
with open('conf_mat_2_XGBoost.pkl','rb') as f:
    x = pickle.load(f)
```

In [31]:
```
x
```

Out[31]:
```
array([[66480,    10,  1437,     0,   479,  4107,  2606],
       [    3, 74796,     8,     0,    63,     0,     0],
       [  747,    40, 62314,     0,  6000,  4864,   955],
       [    0,     0,     1, 74532,     0,     0,     0],
       [  480,   140,  5722,     0, 68107,     0,   790],
       [  621,     0,  1565,     0,     0, 72382,   213],
       [  942,     0,   345,     0,   549,   610, 72233]])
```

In [32]:
```
with open('conf_mat_pct_2_XGBoost.pkl','wb') as f:
    pickle.dump(conf_mat_pct_2, f)
```

In [ ]:
```

```