```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

```
from sklearn import preprocessing
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
```

```
from sklearn.feature_selection import f_regression
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import precision_score, recall_score, f1_score
```

```
import warnings
warnings.filterwarnings('ignore')
```

```
df = pd.read_csv("drug200.csv")
df.head()
```

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	DrugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	DrugY

```
df['Drug'].unique()
```

```
array(['DrugY', 'drugC', 'drugX', 'drugA', 'drugB'], dtype=object)
```

```
df['Cholesterol'].unique()
df["BP"].unique()
df['Sex'].unique()
```

```
array(['F', 'M'], dtype=object)
```

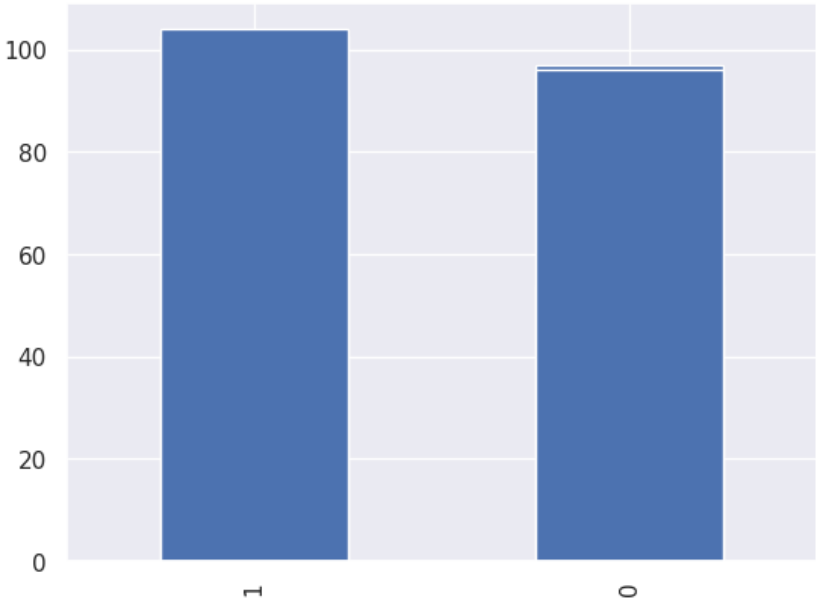
```
le1 = preprocessing.LabelEncoder()
df['Drug']= le1.fit_transform(df['Drug'])
df['Cholesterol']= le1.fit_transform(df['Cholesterol'])
df['BP']= le1.fit_transform(df['BP'])
df['Sex']= le1.fit_transform(df['Sex'])
```

```
len(df)
df["Drug"].value_counts().plot(kind="bar");
df["Cholesterol"].value_counts().plot(kind="bar");
df["Sex"].value_counts().plot(kind="bar");
df.info()
```

```
df.isna().sum()
df.describe()
pd.crosstab(df["Drug"], df["Sex"])
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Age          200 non-null    int64
 1   Sex          200 non-null    int64
 2   BP           200 non-null    int64
 3   Cholesterol  200 non-null    int64
 4   Na_to_K      200 non-null    float64
 5   Drug         200 non-null    int64
dtypes: float64(1), int64(5)
memory usage: 9.5 KB
```

Sex Drug	0	1
0	47	44
1	9	14
2	6	10
3	7	9
4	27	27

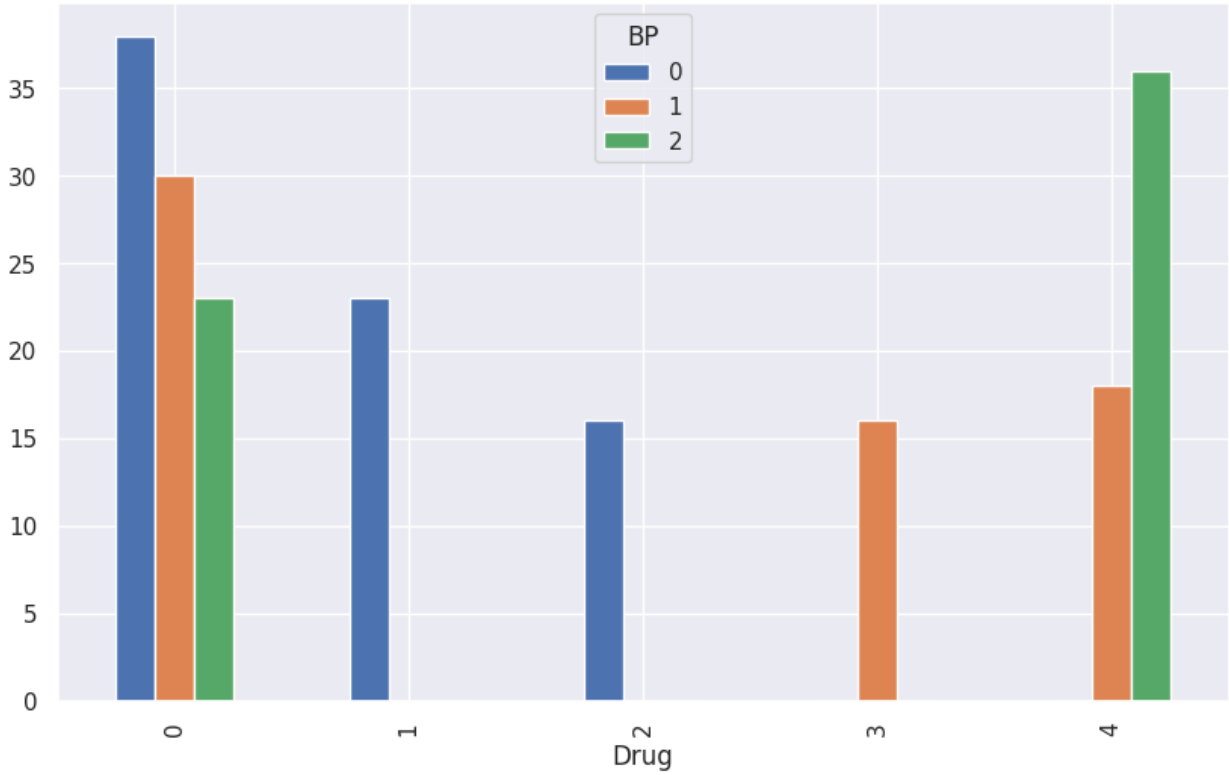



```
pd.crosstab(df["Drug"], df["Sex"]).plot(kind="bar", figsize=(10, 6))
plt.title("Drug vs Sex")
plt.xlabel("Drug Type")
plt.ylabel("Sex")
plt.xticks(rotation=0);
pd.crosstab(df["Drug"], df["BP"])
```

BP	0	1	2
Drug			
0	38	30	23
1	23	0	0
2	16	0	0
3	0	16	0
4	0	18	36

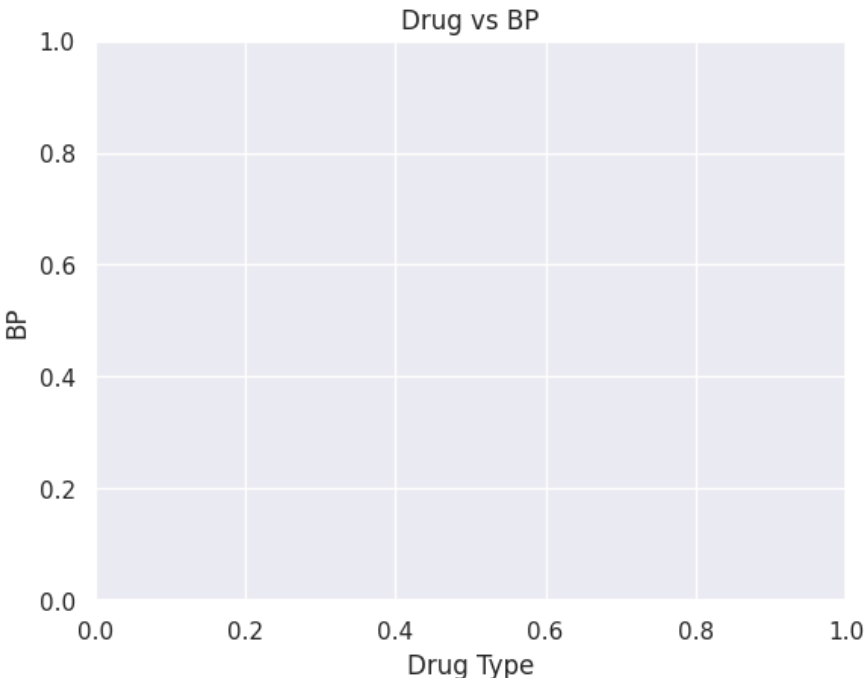

```
pd.crosstab(df["Drug"], df["BP"]).plot(kind="bar",
                                       figsize=(10, 6))
```

```
<Axes: xlabel='Drug'>
```



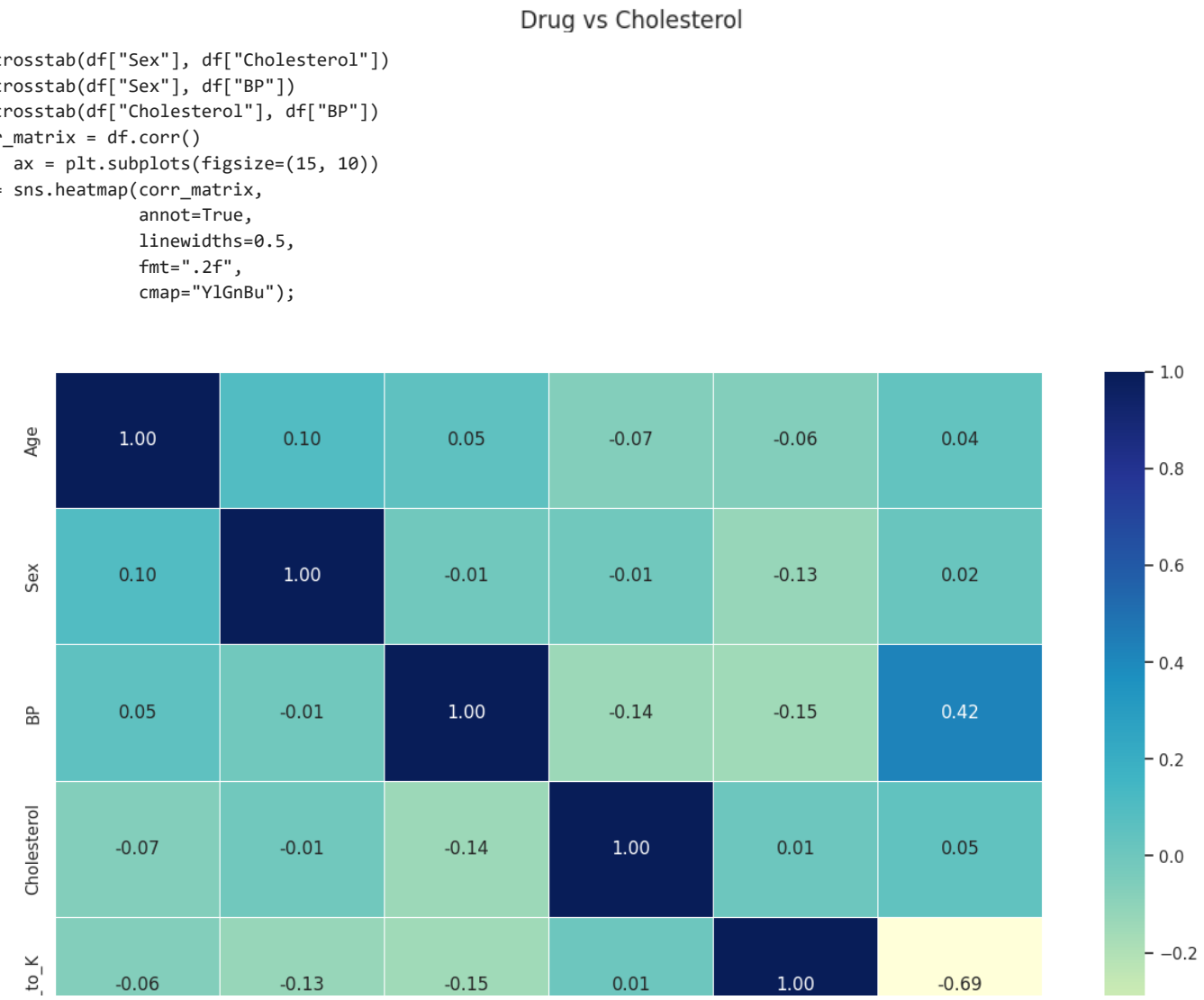

```
plt.title("Drug vs BP")
plt.xlabel("Drug Type")
plt.ylabel("BP")
plt.xticks(rotation=0);
pd.crosstab(df["Drug"], df["Cholesterol"])
```

Cholesterol	0	1
Drug		
0	47	44
1	12	11
2	8	8
3	16	0
4	20	34




```
pd.crosstab(df["Drug"], df["Cholesterol"]).plot(kind="bar",
                                    figsize=(10, 6))
plt.title("Drug vs Cholesterol")
plt.xlabel("Drug Type")
plt.ylabel("Cholesterol")
plt.xticks(rotation=0);
```

```
pd.crosstab(df["Sex"], df["Cholesterol"])
pd.crosstab(df["Sex"], df["BP"])
pd.crosstab(df["Cholesterol"], df["BP"])
corr_matrix = df.corr()
fig, ax = plt.subplots(figsize=(15, 10))
ax = sns.heatmap(corr_matrix,
                 annot=True,
                 linewidths=0.5,
                 fmt=".2f",
                 cmap="YlGnBu");
```




```
np.random.seed(42)
```

```
X_train, X_test, y_train, y_test = train_test_split(df.drop("Drug", axis = 1),
                                                    df["Drug"],
                                                    test_size=0.1)
x = df.drop("Drug", axis = 1)
y = df["Drug"]
```

```
p_values = f_regression(x,y)[1]
len(p_values)
l = []
```

```
col = list(df.columns)
iter_df = 0
```

```
for value in p_values:
    if col[iter_df] == "close":
        iter_df += 1

    l.append({col[iter_df] : value.round(3)})
    iter_df += 1

l
```

```
[{'Age': 0.556},
 {'Sex': 0.798},
```

```
        {'BP': 0.0},
        {'Cholesterol': 0.496},
        {'Na_to_K': 0.0}]
```

```
# Put models in a dictionary
models = {"ANN": MLPClassifier(),
          "Random Forest": RandomForestClassifier(),
          "SVM": SVC(),
         "XGBoost": XGBClassifier()}
```

```
# Create a function to fit and score models
def fit_and_score(models, X_train, X_test, y_train, y_test):
    """
    Fits and evaluates given machine learning models.
    models : a dict of differetn Scikit-Learn machine learning models
    X_train : training data (no labels)
    X_test : testing data (no labels)
    y_train : training labels
    y_test : test labels
    """
    # Set random seed
    np.random.seed(42)
    # Make a dictionary to keep model scores
    model_scores = {}
    # Loop through models
    for name, model in models.items():
        # Fit the model to the data
        model.fit(X_train, y_train)
        # Evaluate the model and append its score to model_scores
        model_scores[name] = model.score(X_test, y_test)
    return model_scores
model_scores = fit_and_score(models=models,
                             X_train=X_train,
                             X_test=X_test,
                             y_train=y_train,
                             y_test=y_test)
```

```
model_scores
```

```
{'ANN': 0.45, 'Random Forest': 1.0, 'SVM': 0.65, 'XGBoost': 1.0}
```

```
model_compare = pd.DataFrame(model_scores, index=["accuracy"])
model_compare.T.plot.bar();
```


```
rf_grid = {"n_estimators": np.arange(10, 1000, 50),
           "max_depth": [None, 3, 5, 10],
           "min_samples_split": np.arange(2, 20, 2),
           "min_samples_leaf": np.arange(1, 20, 2)}
```

```
# Setup random seed
np.random.seed(42)
```

```
# Setup random hyperparameter search for RandomForestClassifier
rs_rf = RandomizedSearchCV(RandomForestClassifier(),
                           param_distributions=rf_grid,
                           cv=5,
                           n_iter=20,
                           verbose=True)
```

```
# Fit random hyperparameter search model for RandomForestClassifier()
rs_rf.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
```

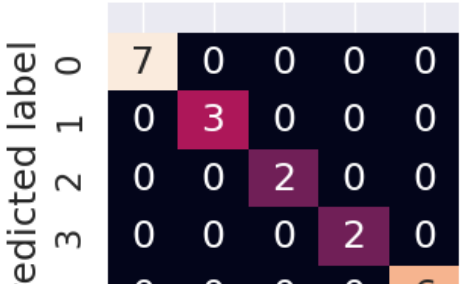
```
▸ RandomizedSearchCV
▸ estimator: RandomForestClassifier
  ▸ RandomForestClassifier
```

```
rs_rf.best_params_
rs_rf.score(X_test, y_test)
y_preds = rs_rf.predict(X_test)
sns.set(font_scale=1.5)
```

```
def plot_conf_mat(y_test, y_preds):
    """
    Plots a nice looking confusion matrix using Seaborn's heatmap()
    """
    fig, ax = plt.subplots(figsize=(3, 3))
    ax = sns.heatmap(confusion_matrix(y_test, y_preds),
                     annot=True,
                     cbar=False)
    plt.xlabel("True label")
    plt.ylabel("Predicted label")

    bottom, top = ax.get_ylim()
    ax.set_ylim(bottom + 0.5, top - 0.5)
```

```
plot_conf_mat(y_test, y_preds)
print(classification_report(y_test, y_preds))
import pickle
from sklearn.ensemble import RandomForestClassifier
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         7
           1       1.00      1.00      1.00         3
           2       1.00      1.00      1.00         2
           3       1.00      1.00      1.00         2
           4       1.00      1.00      1.00         6

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20
```




```
!pip install nbconvert
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: nbconvert in /usr/local/lib/python3.9/dist-packages (6.5.4)
Requirement already satisfied: tinycss2 in /usr/local/lib/python3.9/dist-packages (from nbconvert) (1.2.1)
Requirement already satisfied: jinja2>=3.0 in /usr/local/lib/python3.9/dist-packages (from nbconvert) (3.1.2)
Requirement already satisfied: lxml in /usr/local/lib/python3.9/dist-packages (from nbconvert) (4.9.2)
Requirement already satisfied: nbformat>=5.1 in /usr/local/lib/python3.9/dist-packages (from nbconvert) (5.8.0)
Requirement already satisfied: jupyterlab-pygments in /usr/local/lib/python3.9/dist-packages (from nbconvert) (0.2
Requirement already satisfied: nbclient>=0.5.0 in /usr/local/lib/python3.9/dist-packages (from nbconvert) (0.7.3)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.9/dist-packages (from nbconvert) (4.11.2)
Requirement already satisfied: bleach in /usr/local/lib/python3.9/dist-packages (from nbconvert) (6.0.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.9/dist-packages (from nbconvert) (2.1.2)
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.9/dist-packages (from nbconvert) (1.
Requirement already satisfied: entrypoints>=0.2.2 in /usr/local/lib/python3.9/dist-packages (from nbconvert) (0.4)
Requirement already satisfied: traitlets>=5.0 in /usr/local/lib/python3.9/dist-packages (from nbconvert) (5.7.1)
Requirement already satisfied: mistune<2,>=0.8.1 in /usr/local/lib/python3.9/dist-packages (from nbconvert) (0.8.4
Requirement already satisfied: jupyter-core>=4.7 in /usr/local/lib/python3.9/dist-packages (from nbconvert) (5.3.0
Requirement already satisfied: packaging in /usr/local/lib/python3.9/dist-packages (from nbconvert) (23.1)
Requirement already satisfied: defusedxml in /usr/local/lib/python3.9/dist-packages (from nbconvert) (0.7.1)
Requirement already satisfied: pygments>=2.4.1 in /usr/local/lib/python3.9/dist-packages (from nbconvert) (2.14.0)
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.9/dist-packages (from jupyter-core>=4.7
Requirement already satisfied: jupyter-client>=6.1.12 in /usr/local/lib/python3.9/dist-packages (from nbclient>=0.
Requirement already satisfied: fastjsonschema in /usr/local/lib/python3.9/dist-packages (from nbformat>=5.1->nbcon
Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.9/dist-packages (from nbformat>=5.1->nbco
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.9/dist-packages (from beautifulsoup4->nbcon
Requirement already satisfied: webencodings in /usr/local/lib/python3.9/dist-packages (from bleach->nbconvert) (0.
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.9/dist-packages (from bleach->nbconvert) (1.16
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.9/dist-packages (from jsonschema>=2.6->nbfo
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in /usr/local/lib/python3.9/dist-pack
Requirement already satisfied: tornado>=4.1 in /usr/local/lib/python3.9/dist-packages (from jupyter-client>=6.1.12
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.9/dist-packages (from jupyter-client
Requirement already satisfied: pyzmq>=13 in /usr/local/lib/python3.9/dist-packages (from jupyter-client>=6.1.12->n
```

```
!jupyter nbconvert --to html Throid.ipynb
```

```
[NbConvertApp] Converting notebook Throid.ipynb to html
[NbConvertApp] Writing 860731 bytes to Throid.html
```

✓ 1s completed at 11:04 AM