Heart Disease Classification Problem

In [3]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

In [5]:

```
df = pd.read_csv("heart-disease.csv")
df
```

Out[5]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

In [6]:

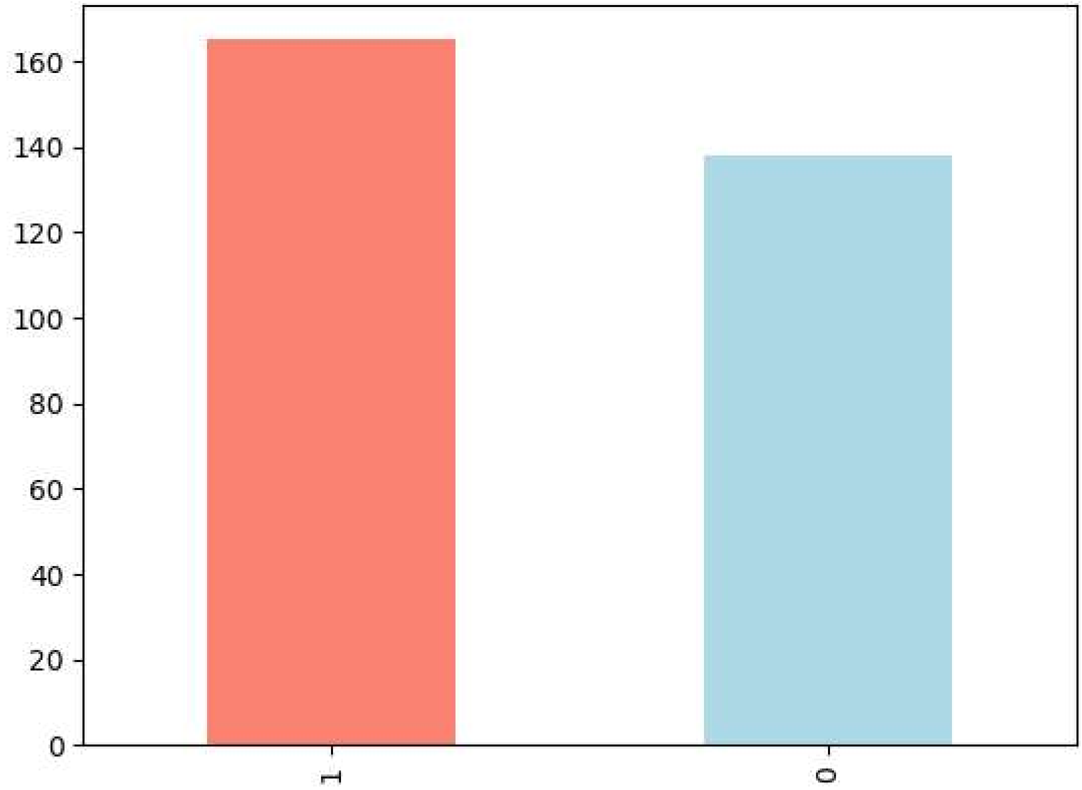
```
df.target.value_counts(normalize=True) * 100
```

Out[6]:

```
1    54.455446
0    45.544554
Name: target, dtype: float64
```

In [7]:

```
df.target.value_counts().plot(kind="bar",color=["salmon", "lightblue"]);
```

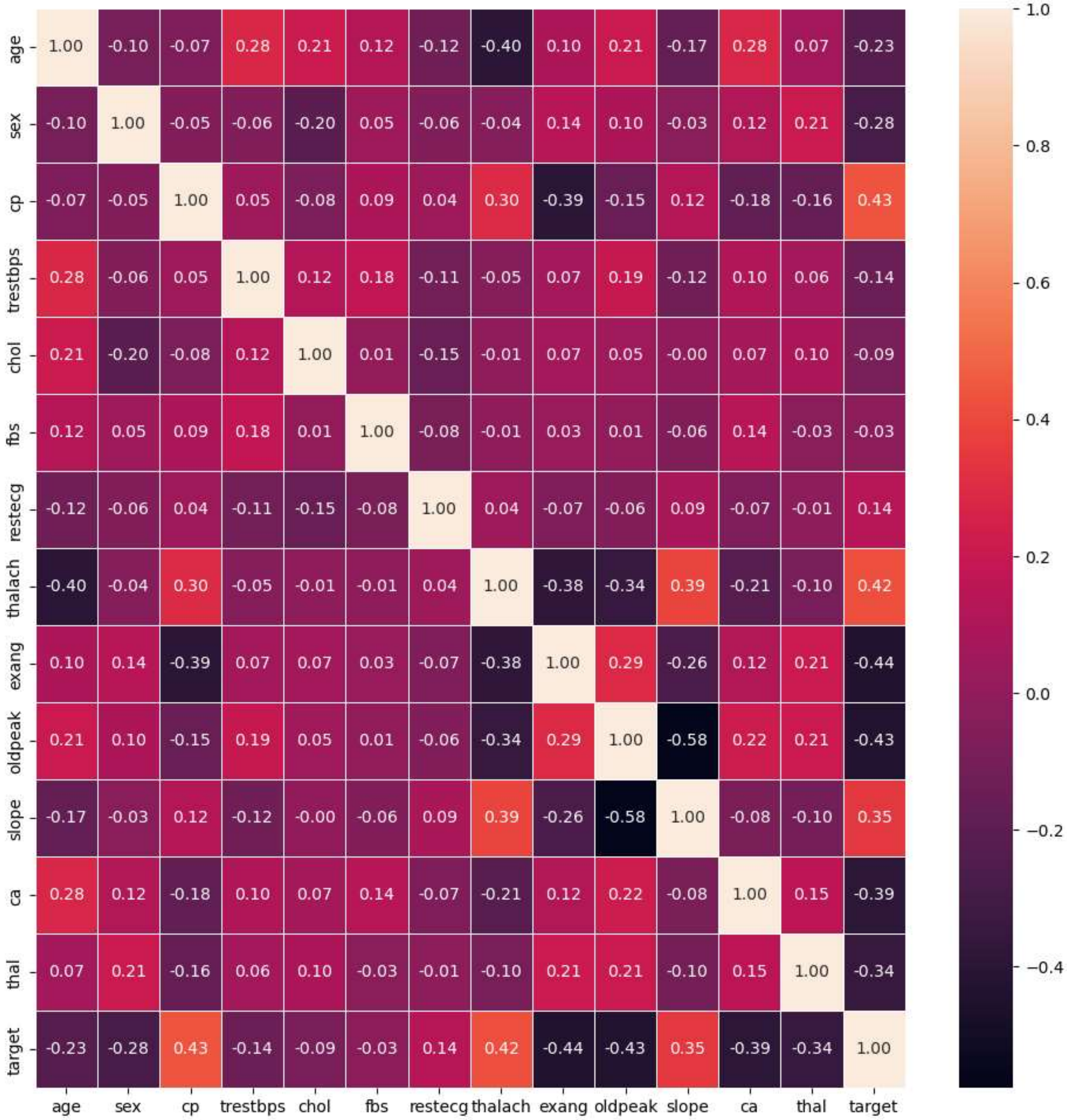


In [6]:
```
pd.crosstab(df.target,df.cp)
```

Out[6]:

cp	0	1	2	3
target				
0	104	9	18	7
1	39	41	69	16

In [7]:
```
plt.figure(figsize=(12, 12))
sns.heatmap(df.corr(),
            annot=True,
            linewidths=0.5,
            fmt= ".2f",
            );
```



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
age	1.00	-0.10	-0.07	0.28	0.21	0.12	-0.12	-0.40	0.10	0.21	-0.17	0.28	0.07	-0.23
sex	-0.10	1.00	-0.05	-0.06	-0.20	0.05	-0.06	-0.04	0.14	0.10	-0.03	0.12	0.21	-0.28
cp	-0.07	-0.05	1.00	0.05	-0.08	0.09	0.04	0.30	-0.39	-0.15	0.12	-0.18	-0.16	0.43
trestbps	0.28	-0.06	0.05	1.00	0.12	0.18	-0.11	-0.05	0.07	0.19	-0.12	0.10	0.06	-0.14
chol	0.21	-0.20	-0.08	0.12	1.00	0.01	-0.15	-0.01	0.07	0.05	-0.00	0.07	0.10	-0.09
fbs	0.12	0.05	0.09	0.18	0.01	1.00	-0.08	-0.01	0.03	0.01	-0.06	0.14	-0.03	-0.03
restecg	-0.12	-0.06	0.04	-0.11	-0.15	-0.08	1.00	0.04	-0.07	-0.06	0.09	-0.07	-0.01	0.14
thalach	-0.40	-0.04	0.30	-0.05	-0.01	-0.01	0.04	1.00	-0.38	-0.34	0.39	-0.21	-0.10	0.42
exang	0.10	0.14	-0.39	0.07	0.07	0.03	-0.07	-0.38	1.00	0.29	-0.26	0.12	0.21	-0.44
oldpeak	0.21	0.10	-0.15	0.19	0.05	0.01	-0.06	-0.34	0.29	1.00	-0.58	0.22	0.21	-0.43
slope	-0.17	-0.03	0.12	-0.12	-0.00	-0.06	0.09	0.39	-0.26	-0.58	1.00	-0.08	-0.10	0.35
ca	0.28	0.12	-0.18	0.10	0.07	0.14	-0.07	-0.21	0.12	0.22	-0.08	1.00	0.15	-0.39
thal	0.07	0.21	-0.16	0.06	0.10	-0.03	-0.01	-0.10	0.21	0.21	-0.10	0.15	1.00	-0.34
target	-0.23	-0.28	0.43	-0.14	-0.09	-0.03	0.14	0.42	-0.44	-0.43	0.35	-0.39	-0.34	1.00

In [8]:

```
x = df.drop("target", axis= 1)
y = df.target
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)
```

In [9]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
models = {"Random Forest":RandomForestClassifier(),"Logistic Regression":LogisticRegres
```

In [10]:

```
def fit_and_score(models,x_train, x_test, y_train, y_test):
    models_scores = {}
    for name,model in models.items():
        model.fit(x_train, y_train)
        models_scores[name] = model.score(x_test, y_test)
    return models_scores
models_scores = fit_and_score(models,x_train, x_test, y_train, y_test)
models_scores
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: Conve
rgenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[10]:

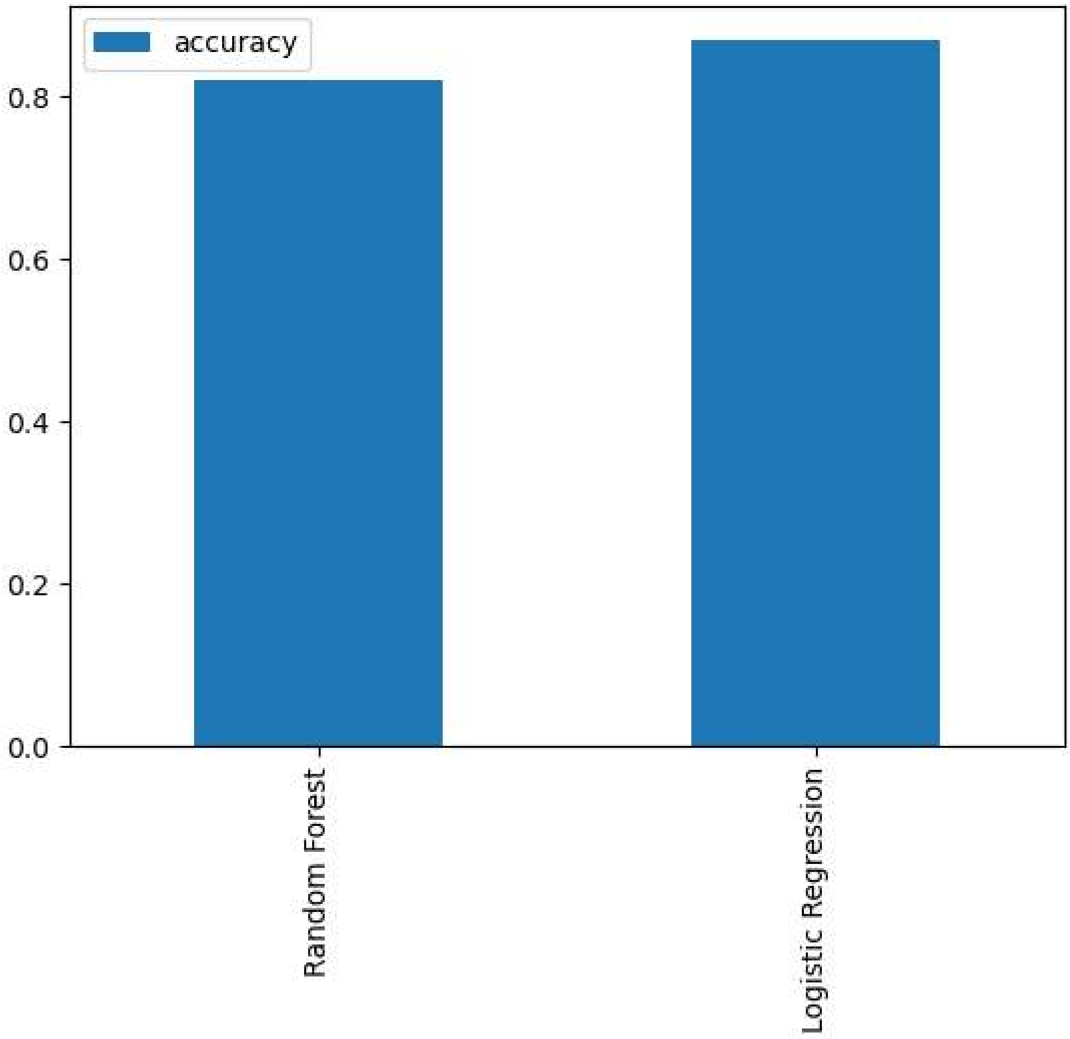
```
{'Random Forest': 0.819672131147541, 'Logistic Regression': 0.8688524590163934}
```

In [11]:

```
models_df = pd.DataFrame(models_scores, index=["accuracy"])
models_df.T.plot(kind="bar")
```

Out[11]:

```
<AxesSubplot:>
```



In [12]:

```
lr_grid = {"C":np.logspace(-4,4,20),"solver":["liblinear"]}
lr_gs = GridSearchCV(LogisticRegression(),lr_grid,cv = 5,verbose=True)
lr_gs.fit(x_train, y_train)
```

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

Out[12]:

```
GridSearchCV(cv=5, estimator=LogisticRegression(),
             param_grid={'C': array([1.00000000e-04, 2.63665090e-04, 6.95192796e-04, 1.
83298071e-03,
       4.83293024e-03, 1.27427499e-02, 3.35981829e-02, 8.85866790e-02,
       2.33572147e-01, 6.15848211e-01, 1.62377674e+00, 4.28133240e+00,
       1.12883789e+01, 2.97635144e+01, 7.84759970e+01, 2.06913808e+02,
       5.45559478e+02, 1.43844989e+03, 3.79269019e+03, 1.00000000e+04]),
                         'solver': ['liblinear']},
             verbose=True)
```

In [13]:

```
lr_gs.best_params_
```

Out[13]:

```
{'C': 0.08858667904100823, 'solver': 'liblinear'}
```

In [14]:

```
lr_gs.score(x_test, y_test)
```

Out[14]:

```
0.8360655737704918
```

In [15]:

```
models_scores
```

Out[15]:

```
{'Random Forest': 0.819672131147541, 'Logistic Regression': 0.8688524590163934}
```

In [16]:

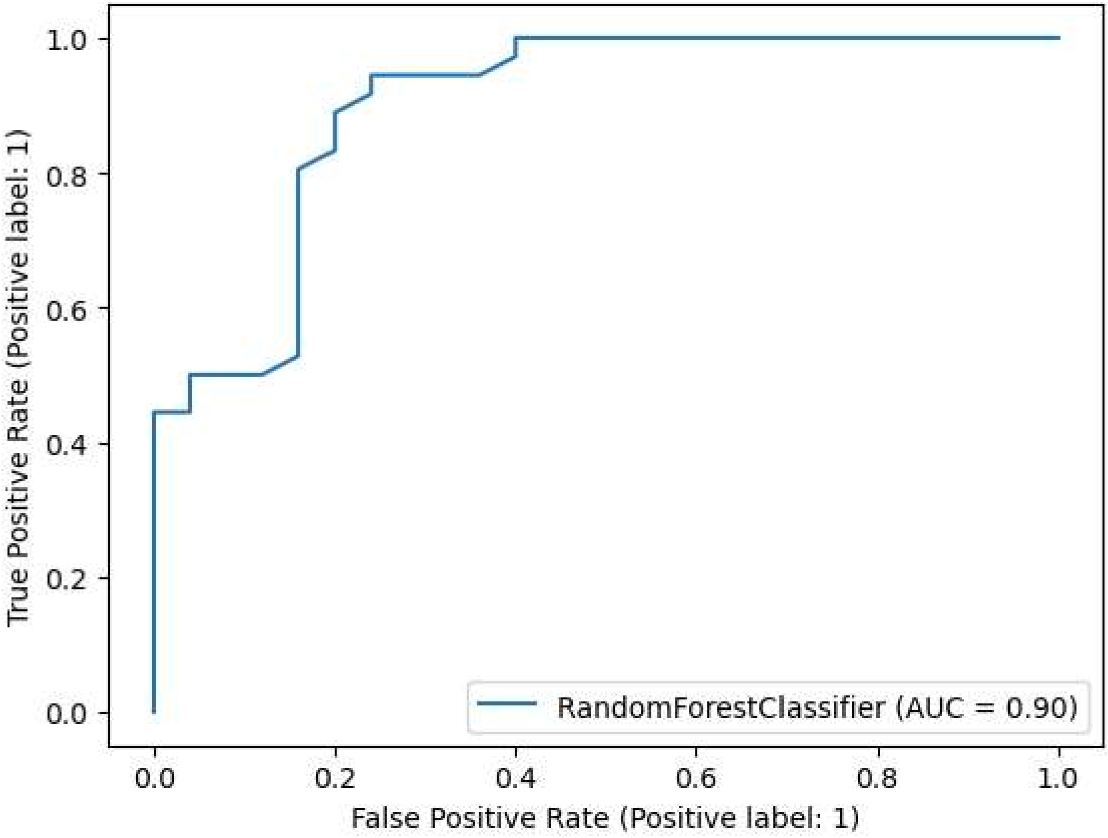
```
model = models["Random Forest"]
```

In [18]:

```
from sklearn.metrics import plot_roc_curve,confusion_matrix

# Plot ROC curve and calculate AUC metric
plot_roc_curve(model, x_test, y_test)
plt.savefig(fname="png.png")
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarni
ng: Function plot_roc_curve is deprecated; Function :func:`plot_roc_curve` is deprecate
d in 1.0 and will be removed in 1.2. Use one of the class methods: :meth:`sklearn.metri
c.RocCurveDisplay.from_predictions` or :meth:`sklearn.metric.RocCurveDisplay.from_estim
ator`.
  warnings.warn(msg, category=FutureWarning)
```



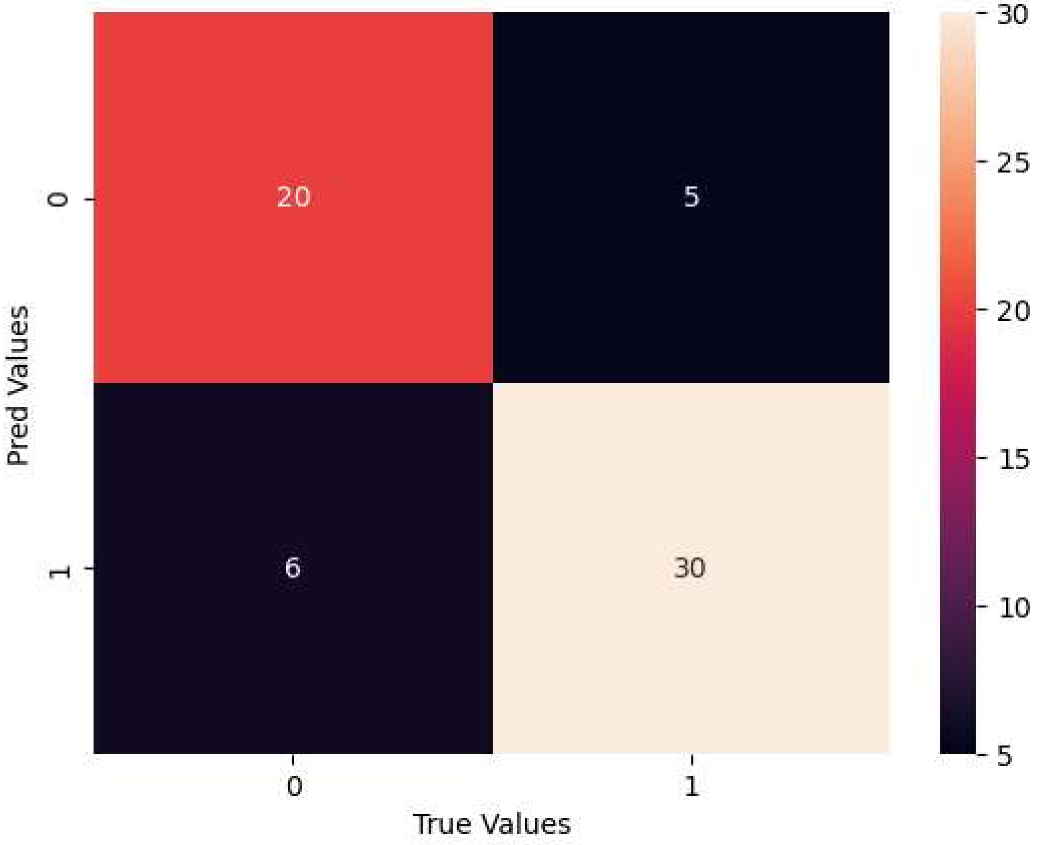
In [23]:
```
y_preds = model.predict(x_test)
y_preds
```

Out[23]:
```
array([0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1,
       1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1], dtype=int64)
```

In [27]:
```
confusion_matrix(y_test, y_preds)
```

Out[27]:
```
array([[20,  5],
       [ 6, 30]], dtype=int64)
```

In [32]:
```
fig, ax = plt.subplots()
ax = sns.heatmap(confusion_matrix(y_test, y_preds),
                annot = True,
                )
plt.xlabel("True Values")
plt.ylabel("Pred Values")
fig.savefig("Confusion Matrix")
```



In [36]:
```
from sklearn.model_selection import cross_val_score

cv_acc = np.mean(cross_val_score(model,
                         x,
                         y,
                         cv=5, # 5-fold cross-validation
                         scoring="accuracy"))
cv_acc
```

Out[36]:
```
0.8182513661202186
```

In [38]:
```
cv_pre = np.mean(cross_val_score(model,
                         x,
                         y,
                         cv=5, # 5-fold cross-validation
                         scoring="precision"))
cv_pre
```

Out[38]:
```
0.8284248053288301
```

In [41]:
```
cv_recall = np.mean(cross_val_score(model,
                         x,
                         y,
                         cv=5, # 5-fold cross-validation
                         scoring="recall"))
cv_recall
```

Out[41]:
```
0.8727272727272727
```

In [43]:

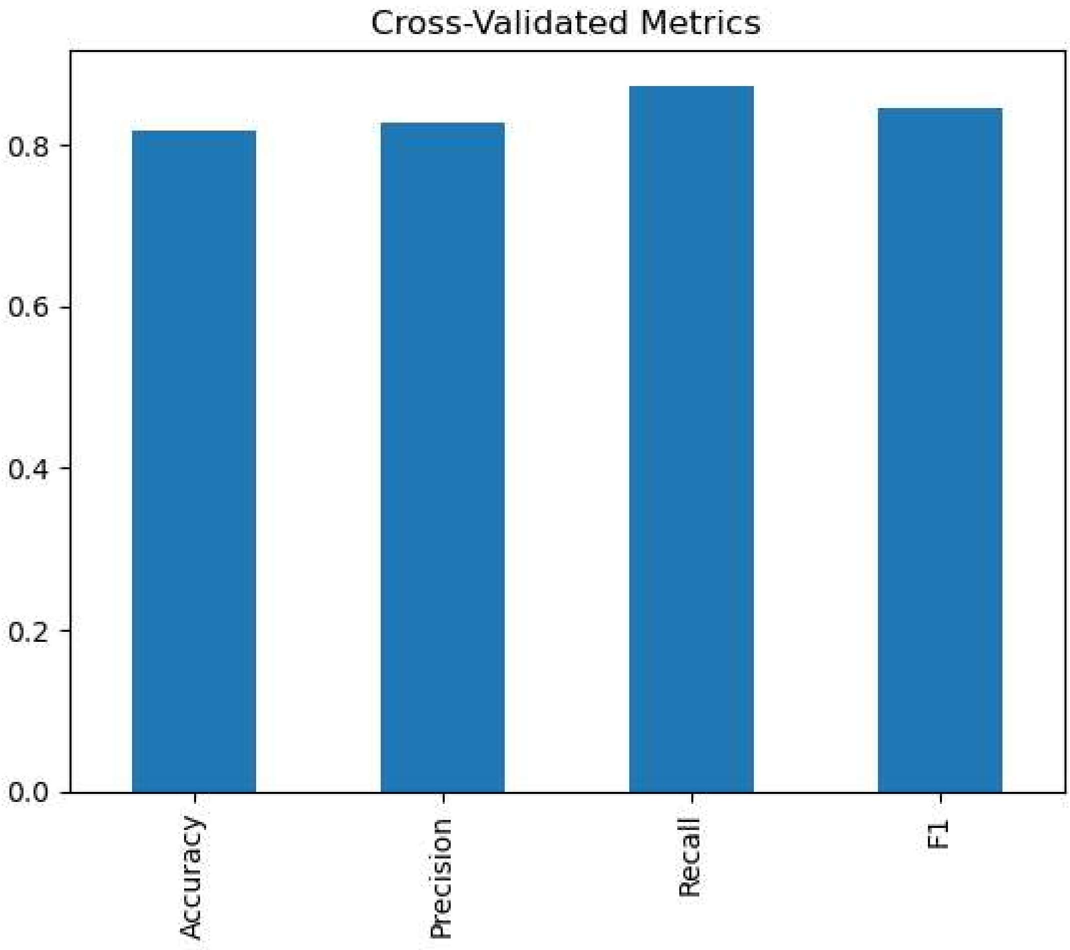
```
cv_f1 = np.mean(cross_val_score(model,
                                x,
                                y,
                                cv=5, # 5-fold cross-validation
                                scoring="f1"))
cv_f1
```

Out[43]:

```
0.8442794468492067
```

In [55]:

```
# Visualizing cross-validated metrics
cv_metrics = pd.DataFrame({"Accuracy": cv_acc,
                            "Precision": cv_pre,
                            "Recall": cv_recall,
                            "F1": cv_f1}, index=[0])
cv_metrics.T.plot(kind="bar",title="Cross-Validated Metrics", legend=False);
plt.savefig("Cross-Validated Metrics")
```


Cross-Validated Metrics
Accuracy
Precision
Recall
F1