In [2]:
```
import pandas as pd
df = pd.read_csv('dataset.csv')
```

In [3]:
```
df
```

Out[3]:

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

303 rows × 14 columns

Q1. Preprocess the dataset by handling missing values, encoding categorical variables, and scaling the numerical features if necessary.

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

we can see that no missing value.

Also no categorical value.

No need to scaling the numerical features.

Q2. Split the dataset into a training set (70%) and a test set (30%).

In [5]:
```
x = df.drop('target',axis = 1)
y = df['target']
```

In [6]:
```
from sklearn.model_selection import train_test_split

x_test,x_train,y_test,y_train = train_test_split(x,y , test_size = 0.3 , r
x_test.shape,x_train.shape ,y_test.shape,y_train.shape
```

Out[6]:
```
((212, 13), (91, 13), (212,), (91,))
```

Q3. Train a random forest classifier on the training set using 100 trees and a maximum depth of 10 for each tree. Use the default values for other hyperparameters.

In [14]:
```
from sklearn.ensemble import RandomForestClassifier
cl = RandomForestClassifier(n_estimators=100, max_depth= 10)
cl.fit(x_train,y_train)
y_pred = cl.predict(x_test)
```

In [15]:
```
y_pred
```

Out[15]:
```
array([1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1,
       0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0,
       1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1,
       1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0,
       1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1], dtype=int64)
```

Q4. Evaluate the performance of the model on the test set using accuracy, precision, recall, and F1 score.

In [9]:
```
from sklearn.metrics import accuracy_score,classification_report,confusion
print(accuracy_score(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

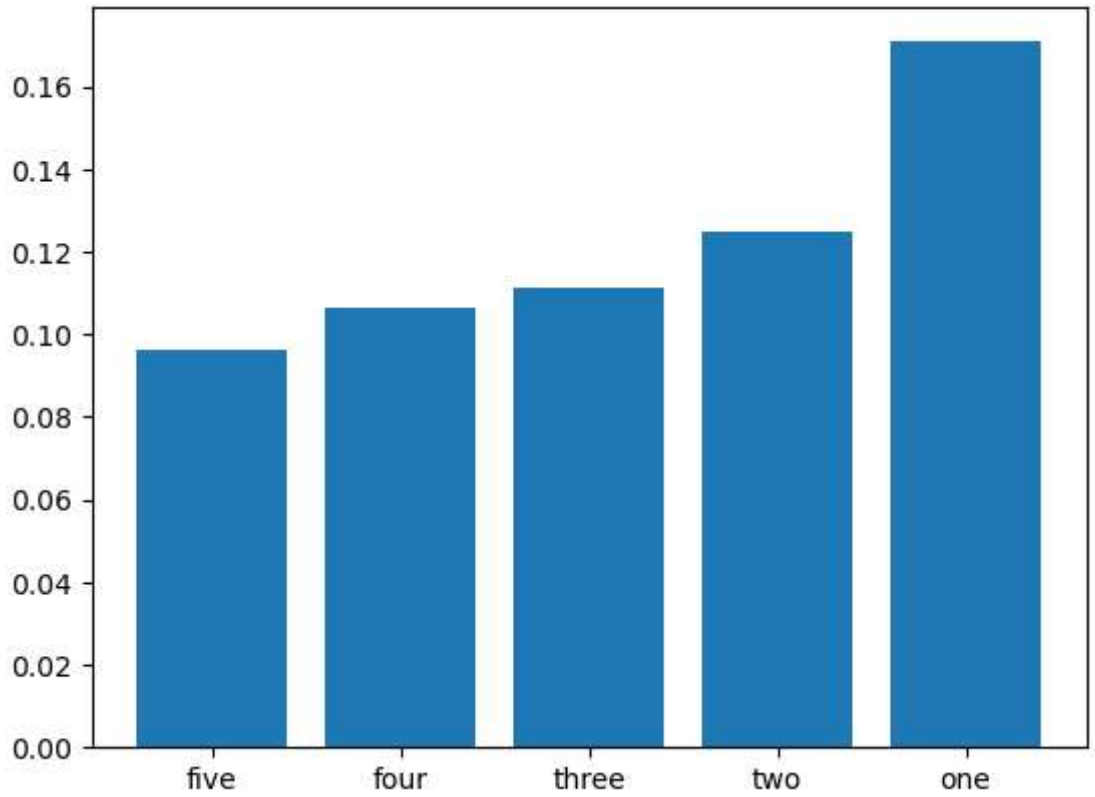
```
0.7830188679245284
              precision    recall  f1-score   support

           0       0.84      0.65      0.73        97
           1       0.75      0.90      0.82       115

    accuracy                           0.78       212
   macro avg       0.80      0.77      0.78       212
weighted avg       0.79      0.78      0.78       212
```

Q5. Use the feature importance scores to identify the top 5 most important features in predicting heart disease risk. Visualise the feature importances using a bar chart.

In [10]:
```
h = cl.feature_importances_
h.sort()
h
```

Out[10]:
```
array([0.00998805, 0.01914236, 0.02962889, 0.03298945, 0.03610461,
       0.0654165 , 0.07087391, 0.07370254, 0.09721412, 0.12611504,
       0.13982773, 0.1435506 , 0.15544619])
```

In [11]:
```
top_5 = [0.09618897, 0.10620078, 0.11145195, 0.12481023, 0.17082303]
x_axis=['five','four','three','two','one']
from matplotlib import pyplot as plt
plt.bar(x_axis,top_5)
plt.show()
```



Q6. Tune the hyperparameters of the random forest classifier using grid search or random search. Try different values of the number of trees, maximum depth, minimum samples split, and minimum samples

In [11]:
```
param = {
    'n_estimators' : [50,70,100,130],
    'criterion' : ['gini','entropy','log_loss'],
    'max_depth' : [5,10,15],
    'min_samples_split' :[2,3,4],
    'min_samples_leaf' :[1,2]
}

from sklearn.model_selection import GridSearchCV
model = GridSearchCV(RandomForestClassifier(),cv = 5,param_grid = param)
model.fit(x_train,y_train)
```

Out[11]:

▸ **GridSearchCV**

▸ **estimator: RandomForestClassifier**

▸ RandomForestClassifier

Q7. Report the best set of hyperparameters found by the search and the corresponding performance metrics. Compare the performance of the tuned model with the default model.

In [16]:
```
cl.best_params_classifier = RandomForestClassifier(criterion = 'entropy',m
```

In [19]:
```
cl.best_params_classifier.fit(x_train,y_train)
y_hyper_pred = cl.best_params_classifier.predict(x_test)
y_hyper_pred
```

Out[19]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1,
       0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0,
       1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1,
       1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1], dtype=int64)
```

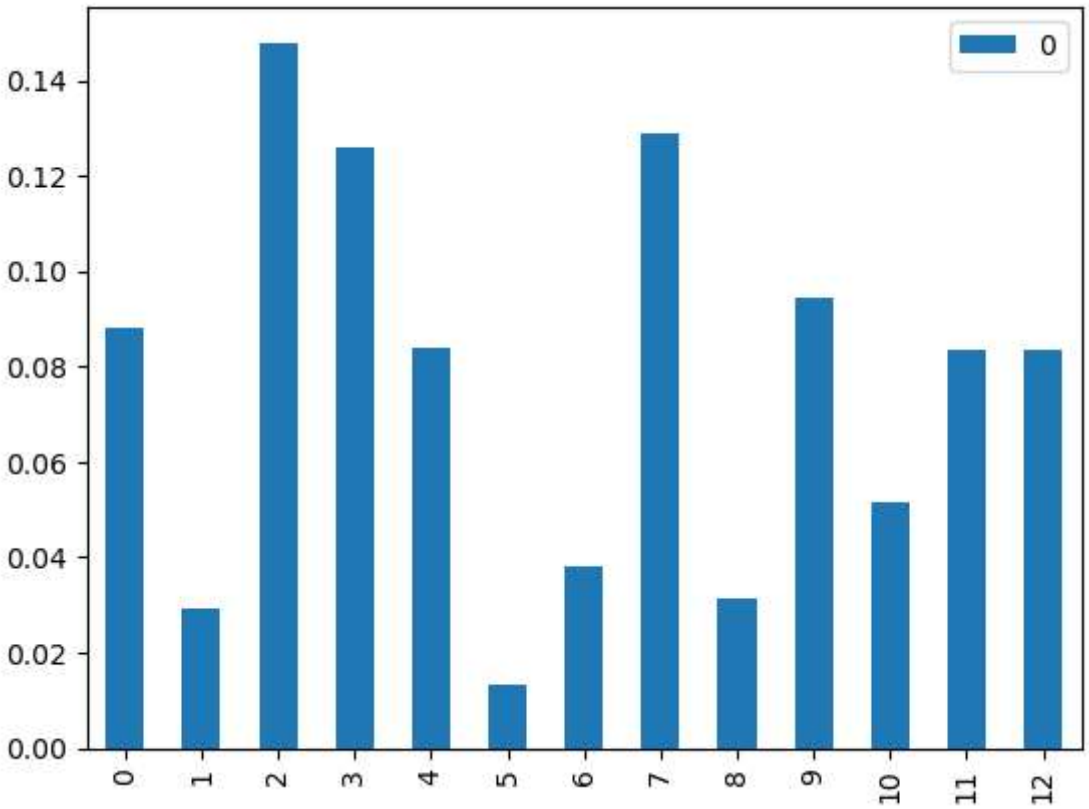
Q8. Interpret the model by analysing the decision boundaries of the random forest classifier. Plot the decision boundaries on a scatter plot of two of the most important features. Discuss the insights and limitations of the model for predicting heart disease risk.

In [20]:
```
print(f'Defualt Model Accuracy : {accuracy_score(y_test,y_pred)}')
print(f'HyperTuned Model Accuracy : {accuracy_score(y_test,y_hyper_pred)}'
```

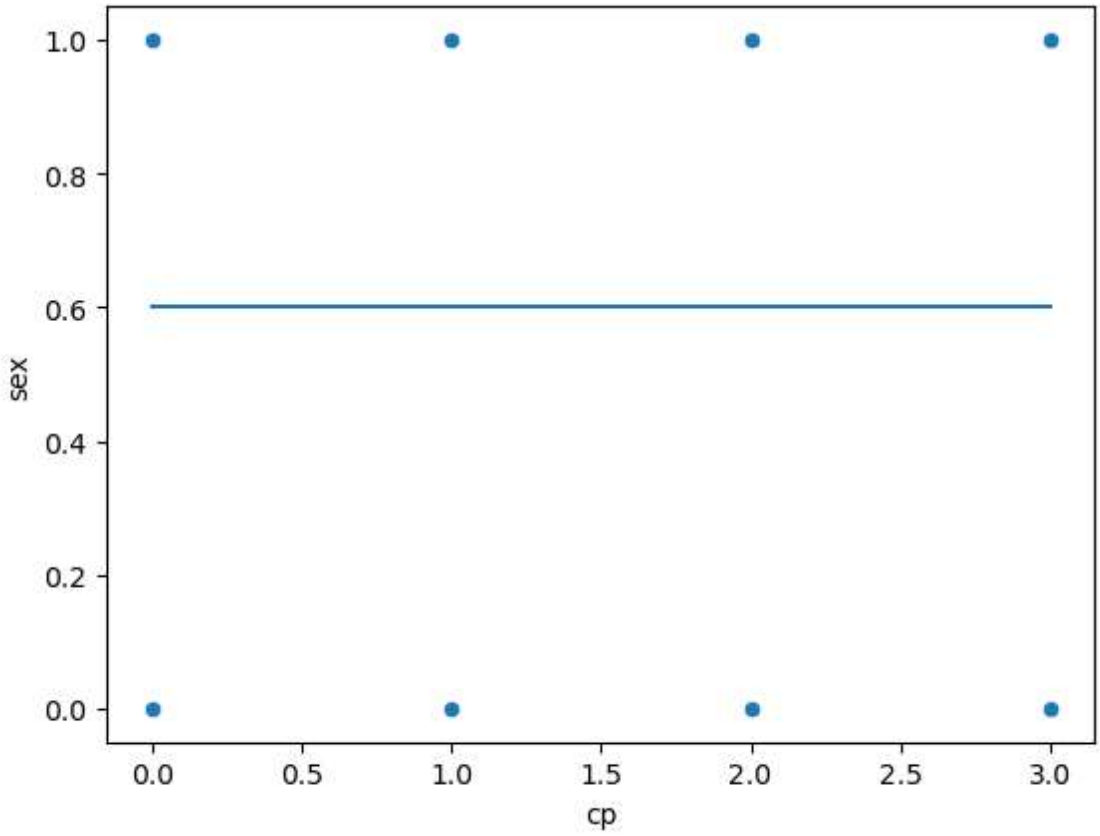
```
Defualt Model Accuracy : 0.7877358490566038
HyperTuned Model Accuracy : 0.7924528301886793
```

In [22]:
```
importance = cl.best_params_classifier.feature_importances_
importance_df = pd.DataFrame(importance)
importance_df.plot.bar()
```

Out[22]: <Axes: >



In [25]:
```
import seaborn as sns
sns.scatterplot(data = df,x = df.iloc[:,2],y = df.iloc[:,1])
plt.plot([0.0,3.0],[0.6,0.6])
plt.show()
```



In [ ]:
```

```