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

```
malaria = pd.read_csv('/content/MalariaData.csv')
```

```
malaria.head()
```

	age	sex	fever	cold	rigor	fatigue	headace	bitter_tongue	vomitting	diarrhea
0	0	Male	yes	no	no	no	no	yes	no	no
1	1	Male	no	no	yes	yes	yes	no	no	no
2	0	Female	yes	no	no	yes	no	yes	no	yes
3	1	Female	yes	yes	yes	no	yes	no	no	no
4	1	Female	yes	yes	yes	no	yes	no	no	yes

```
malaria.shape
```

```
(1000, 18)
```

```
malaria.describe()
```

	age
count	1000.000000
mean	0.512000
std	0.500106
min	0.000000
25%	0.000000
50%	1.000000
75%	1.000000
max	1.000000

```
values = {"fever":{"no":0, "yes":1},"cold":{"no":0, "yes":1},"rigor":{"no":0, "yes":1},"fatig
          "headace":{"no":0, "yes":1},"bitter_tongue":{"no":0, "yes":1},"vomitting":{"no":0,
          "diarrhea":{"no":0, "yes":1},"Convulsion":{"no":0, "yes":1},"Anemia":{"no":0, "yes"
          "cocacola_urine":{"no":0, "yes":1},"hypoglycemia":{"no":0, "yes":1},"prostraction":
```

```
¨nyperpyrexıa¨:{¨no¨:0, ¨yes¨:1},¨nyperpyrexıa¨:{¨no¨:0, ¨yes¨:1},¨severe_malerıa¨:
```

```
malaria.replace(values,inplace=True)
```

```
malaria.head()
```

	age	sex	fever	cold	rigor	fatigue	headace	bitter_tongue	vomitting	diarrhea
0	0	Male	1	0	0	0	0	1	0	0
1	1	Male	0	0	1	1	1	0	0	0
2	0	Female	1	0	0	1	0	1	0	1
3	1	Female	1	1	1	0	1	0	0	0
4	1	Female	1	1	1	0	1	0	0	1

```
y = malaria['severe_maleria']
y
```

```
0      0
1      0
2      0
3      0
4      0
      ..
995    0
996    0
997    0
998    1
999    0
Name: severe_maleria, Length: 1000, dtype: int64
```

```
new_malaria = malaria.drop('severe_maleria', axis=1)
```

```
new_malaria.head()
```

	age	sex	fever	cold	rigor	fatigue	headace	bitter_tongue	vomitting	diarrhea
0	0	Male	1	0	0	0	0	1	0	0
1	1	Male	0	0	1	1	1	0	0	0
2	0	Female	1	0	0	1	0	1	0	1
3	1	Female	1	1	1	0	1	0	0	0
4	1	Female	1	1	1	0	1	0	0	1

```
new_malaria = pd.get_dummies(new_malaria)
```

```
new_malaria = new_malaria.drop('age', axis=1)
```

```
new_malaria.head()
```

	fever	cold	rigor	fatigue	headace	bitter_tongue	vomitting	diarrhea	Convulsion
0	1	0	0	0	0	1	0	0	0
1	0	0	1	1	1	0	0	0	0
2	1	0	0	1	0	1	0	1	1
3	1	1	1	0	1	0	0	0	0
4	1	1	1	0	1	0	0	1	1

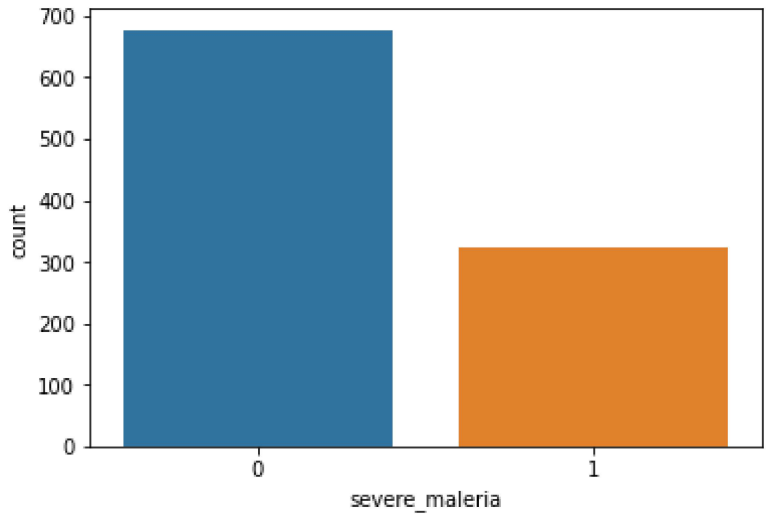
```
print(malaria.groupby('severe_maleria').size())
```

```
severe_maleria
0    677
1    323
dtype: int64
```

```
import seaborn as sns
```

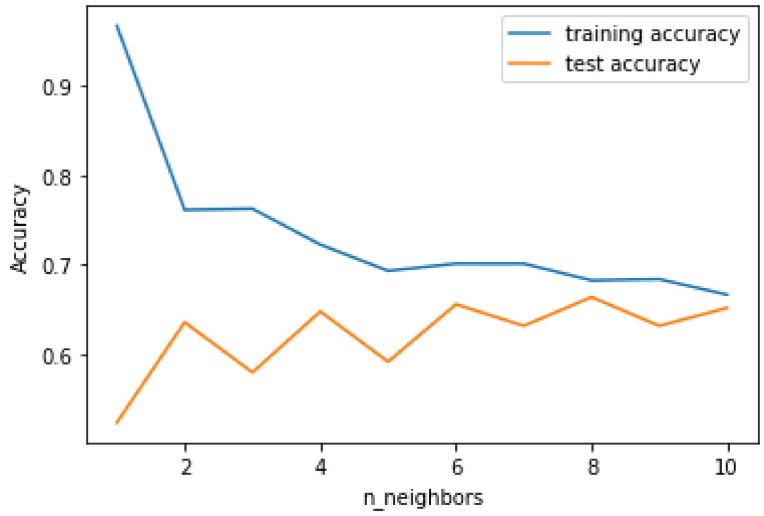
```
sns.countplot(malaria['severe_maleria'],label="Count")
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass th
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f08d48c6550>
```




```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(new_malaria, y, test_size=0.25, random_st


from sklearn.neighbors import KNeighborsClassifier

training_accuracy = []
test_accuracy = []
# try n_neighbors from 1 to 10
neighbors_settings = range(1, 11)

for n_neighbors in neighbors_settings:
    # build the model
    knn = KNeighborsClassifier(n_neighbors=n_neighbors)
    knn.fit(X_train, y_train)
    # record training set accuracy
    training_accuracy.append(knn.score(X_train, y_train))
    # record test set accuracy
    test_accuracy.append(knn.score(X_test, y_test))

plt.plot(neighbors_settings, training_accuracy, label="training accuracy")
plt.plot(neighbors_settings, test_accuracy, label="test accuracy")
plt.ylabel("Accuracy")
plt.xlabel("n_neighbors")
plt.legend()
plt.savefig('knn_compare_model')
```




```
knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(X_train, y_train)

print('Accuracy of K-NN classifier on training set: {:.2f}'.format(knn.score(X_train, y_train
print('Accuracy of K-NN classifier on test set: {:.2f}'.format(knn.score(X_test, y_test)))
```

```
Accuracy of K-NN classifier on training set: 0.67
Accuracy of K-NN classifier on test set: 0.65
```

```
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression(C=1).fit(X_train, y_train)
print("Training set accuracy: {:.3f}".format(logreg.score(X_train, y_train)))
print("Test set accuracy: {:.3f}".format(logreg.score(X_test, y_test)))
```

```
Training set accuracy: 0.667
Test set accuracy: 0.708
```

```
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression(C=0.01).fit(X_train, y_train)
print("Training set accuracy: {:.3f}".format(logreg.score(X_train, y_train)))
print("Test set accuracy: {:.3f}".format(logreg.score(X_test, y_test)))
```

```
Training set accuracy: 0.667
Test set accuracy: 0.708
```

```
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression(C=20).fit(X_train, y_train)
print("Training set accuracy: {:.3f}".format(logreg.score(X_train, y_train)))
print("Test set accuracy: {:.3f}".format(logreg.score(X_test, y_test)))
```

```
Training set accuracy: 0.667
Test set accuracy: 0.708
```

```
from sklearn.tree import DecisionTreeClassifier

tree = DecisionTreeClassifier(random_state=0)
tree.fit(X_train, y_train)
print("Accuracy on training set: {:.3f}".format(tree.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(tree.score(X_test, y_test)))
```

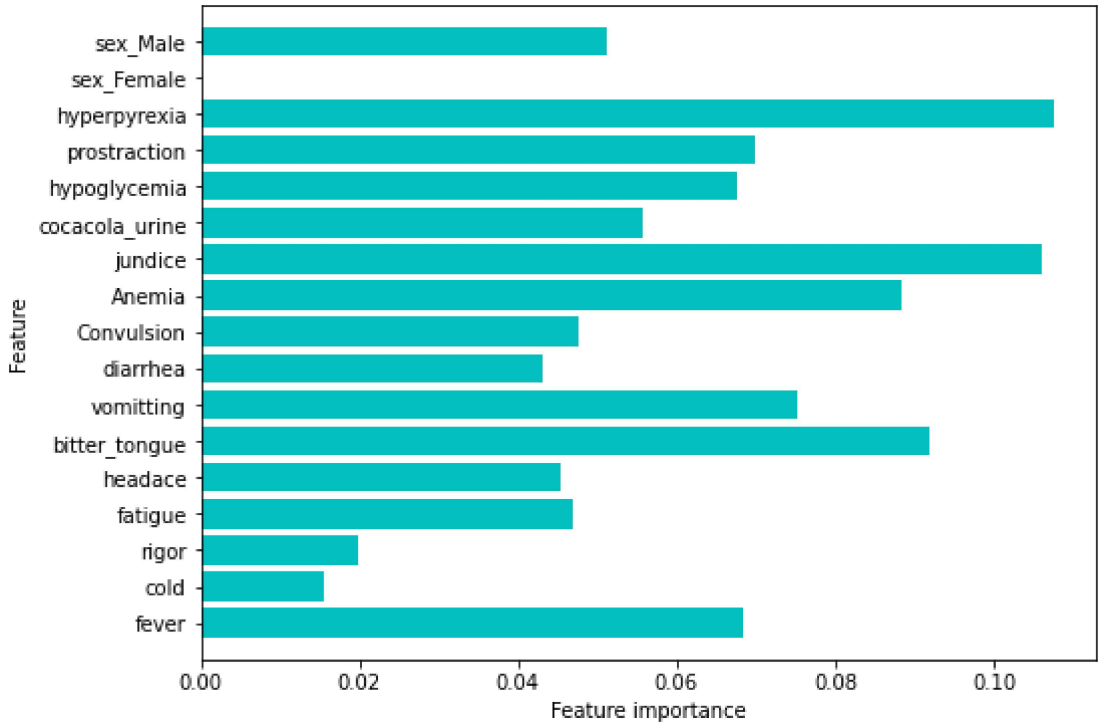
```
Accuracy on training set: 0.968
Accuracy on test set: 0.596
```

```
tree = DecisionTreeClassifier(max_depth=6, random_state=0)
tree.fit(X_train, y_train)

print("Accuracy on training set: {:.3f}".format(tree.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(tree.score(X_test, y_test)))
```

```
Accuracy on training set: 0.723
Accuracy on test set: 0.692
```

```
malaria_features = [x for i,x in enumerate(new_malaria.columns) if i!=17]
def plot_feature_importances_malaria(model):
    plt.figure(figsize=(8,6))
    n_features = 17
```

```
    plt.barh(range(n_features), model.feature_importances_, align='center',color="c")
    plt.yticks(np.arange(n_features), malaria_features)
    plt.xlabel("Feature importance")
    plt.ylabel("Feature")
    plt.ylim(-1, n_features)

plot_feature_importances_malaria(tree)
plt.savefig('feature_importance')
```



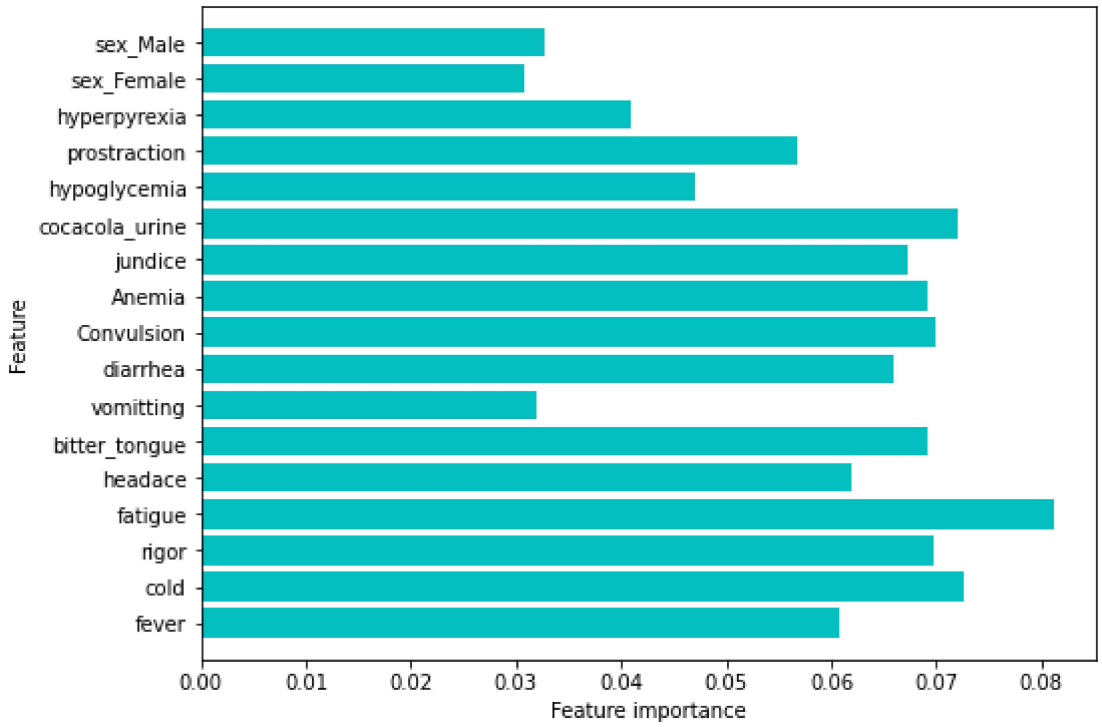

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=100, random_state=0)
rf.fit(X_train, y_train)
print("Accuracy on training set: {:.3f}".format(rf.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(rf.score(X_test, y_test)))
```

```
    Accuracy on training set: 0.968
    Accuracy on test set: 0.636
```

```
rf1 = RandomForestClassifier(max_depth=3, n_estimators=100, random_state=0)
rf1.fit(X_train, y_train)
print("Accuracy on training set: {:.3f}".format(rf1.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(rf1.score(X_test, y_test)))
```

```
    Accuracy on training set: 0.667
    Accuracy on test set: 0.708
```

```
plot_feature_importances_malaria(rf)
```



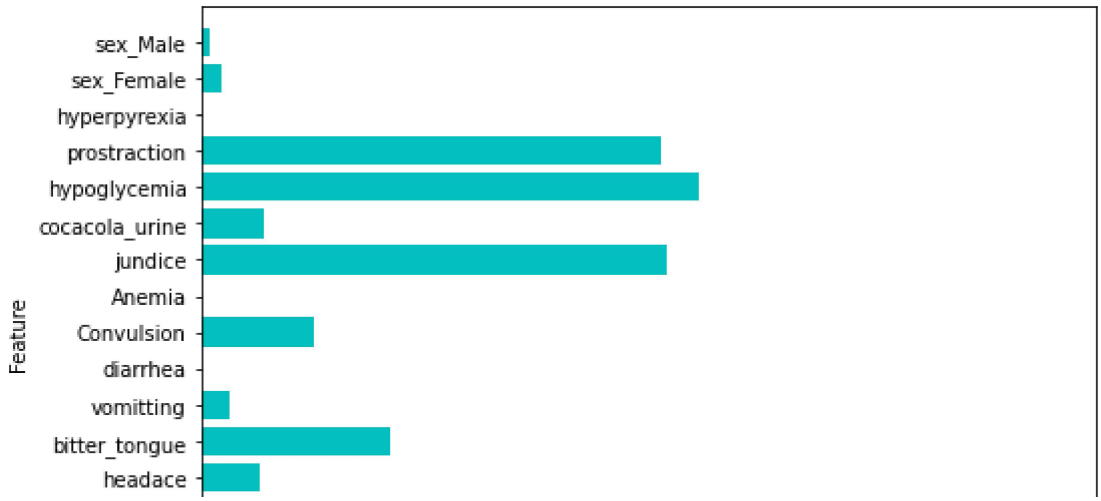

```
from sklearn.ensemble import GradientBoostingClassifier

gb = GradientBoostingClassifier(random_state=0)
gb.fit(X_train, y_train)

print("Accuracy on training set: {:.3f}".format(gb.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(gb.score(X_test, y_test)))
```

```
Accuracy on training set: 0.720
Accuracy on test set: 0.688
```

```
gb1 = GradientBoostingClassifier(random_state=0, max_depth=1)
gb1.fit(X_train, y_train)

print("Accuracy on training set: {:.3f}".format(gb1.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(gb1.score(X_test, y_test)))
```

```
Accuracy on training set: 0.667
Accuracy on test set: 0.708
```

```
plot_feature_importances_malaria(gb1)
```




```
from sklearn.svm import SVC

svc = SVC()
svc.fit(X_train, y_train)

print("Accuracy on training set: {:.2f}".format(svc.score(X_train, y_train)))
print("Accuracy on test set: {:.2f}".format(svc.score(X_test, y_test)))
```

```
Accuracy on training set: 0.67
Accuracy on test set: 0.71
```

✓ 0s completed at 12:58 PM