untitled3

April 23, 2023

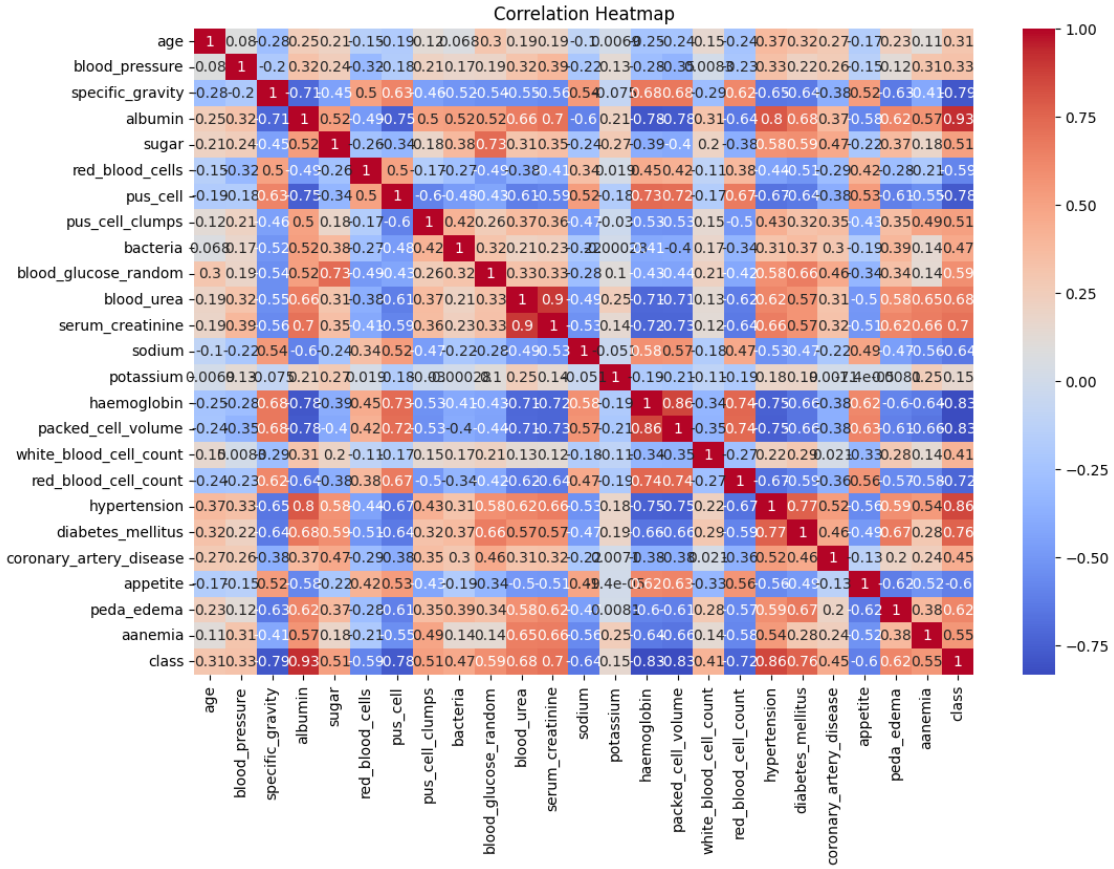
[154]:
```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, GridSearchCV,␣
 ↪cross_val_score, RepeatedStratifiedKFold, StratifiedKFold
from sklearn.metrics import accuracy_score, confusion_matrix,roc_curve,␣
 ↪roc_auc_score, precision_score, recall_score, precision_recall_curve
from sklearn.metrics import f1_score
import seaborn as sns
import matplotlib.pyplot as plt
```

[155]:
```
df = pd.read_csv('kidney_disease.csv')
```

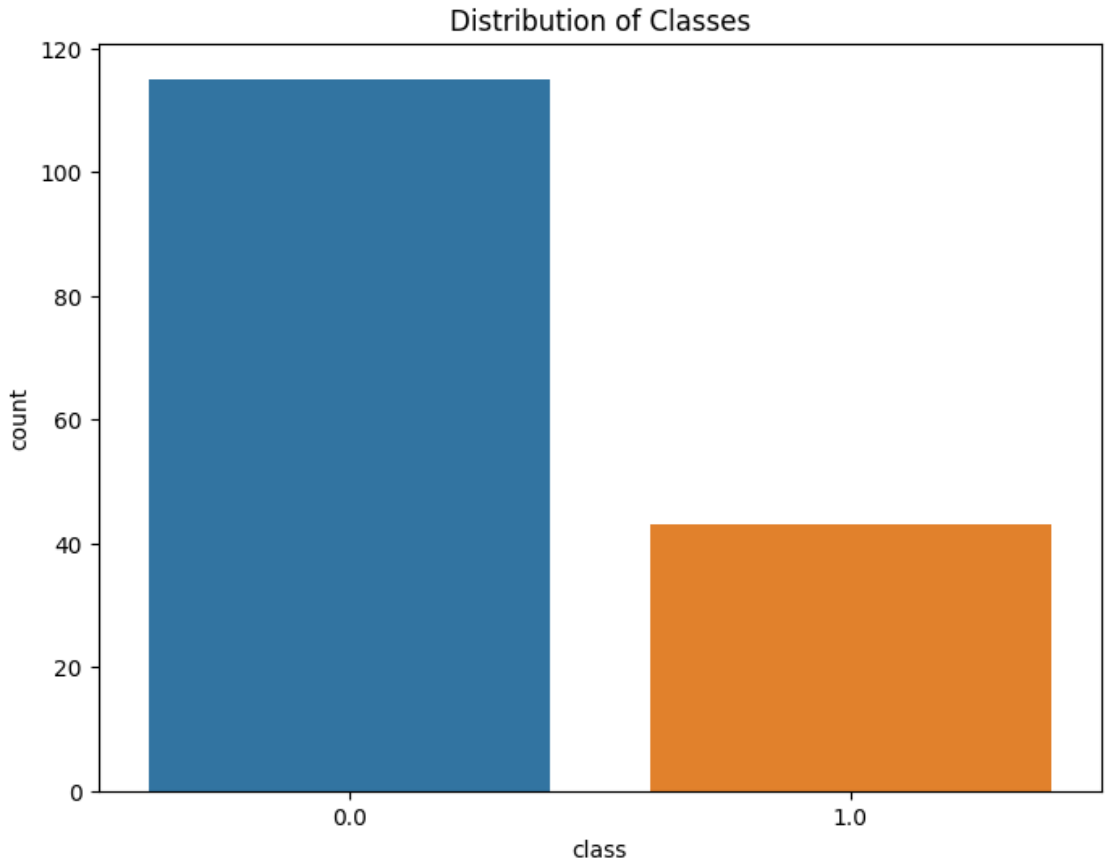
[156]:
```
df.drop('id', axis = 1, inplace = True)
df = df.dropna(how='any')
```

[162]:
```
df.columns = ['age', 'blood_pressure', 'specific_gravity', 'albumin', 'sugar',␣
 ↪'red_blood_cells', 'pus_cell',
              'pus_cell_clumps', 'bacteria', 'blood_glucose_random',␣
 ↪'blood_urea', 'serum_creatinine', 'sodium',
              'potassium', 'haemoglobin', 'packed_cell_volume',␣
 ↪'white_blood_cell_count', 'red_blood_cell_count',
              'hypertension', 'diabetes_mellitus', 'coronary_artery_disease',␣
 ↪'appetite', 'peda_edema',
              'aanemia', 'class']

df = df.replace(to_replace = {'normal': 1, 'abnormal': 0,'good': 1, 'poor' :
 ↪0,'present':1,'notpresent':0,'yes':1,'no':0,'ckd':1,'notckd':0})
df = df.astype(float)
```

[163]:
```
corr = df.corr()

# Plot the correlation heatmap
plt.figure(figsize=(12,8))
sns.heatmap(corr, annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```



[164]:
```
plt.figure(figsize=(8,6))
sns.countplot(x="class", data=df)
plt.title("Distribution of Classes")
plt.show()
```



[166]:
```
y = df['class']
x = df.drop(['class'], axis = 1)
```

[167]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
↪random_state=14)
```

[168]:
```
lg1 = LogisticRegression(random_state=14,max_iter=1000)

lg1.fit(x_train,y_train)

y_pred = lg1.predict(x_test)
print(f'Accuracy Score: {accuracy_score(y_test,y_pred)}')
print(f'Area Under Curve: {roc_auc_score(y_test, y_pred)}')
print(f'Recall score: {recall_score(y_test,y_pred)}')
```

```
Accuracy Score: 0.96875
Area Under Curve: 0.95
Recall score: 0.9
```

[169]:
```
mean = np.mean(x, axis=0)
std = np.std(x, axis=0)

# Normalize the matrix
x_norm = (x - mean) / std

x_train_norm, x_test_norm, y_train_norm, y_test_norm = train_test_split(x_norm,␣
 ↪y, test_size=0.2, random_state=14)

lg1_norm = LogisticRegression(random_state=14, max_iter=1000)

lg1_norm.fit(x_train_norm,y_train_norm)


y_pred_norm = lg1_norm.predict(x_test_norm)
print(f'Accuracy Score: {accuracy_score(y_test_norm,y_pred_norm)}')
print(f'Area Under Curve: {roc_auc_score(y_test_norm, y_pred_norm)}')
print(f'Recall score: {recall_score(y_test_norm,y_pred_norm)}')
```

```
Accuracy Score: 1.0
Area Under Curve: 1.0
Recall score: 1.0
```