```
# In[1]:


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


# In[2]:


data = pd.read_csv("C:/Users/Nabee/Downloads/Data-Science-Capstone-
Projects-master/Data-Science-Capstone-Projects-master/health care
diabetes.csv")


# In[3]:


data.head()


# In[4]:


data.describe()


# In[5]:


data.info()


# In[6]:


data.isnull().any()


# In[7]:


cols_with_null_as_zero = ['Glucose', 'BloodPressure', 'SkinThickness',
'Insulin', 'BMI']


# In[8]:


data[cols_with_null_as_zero] =
data[cols_with_null_as_zero].replace(0,np.NaN)


# In[9]:
```

```python
data.isnull().sum()


# In[10]:


sns.histplot(data['Glucose'])


# In[11]:


sns.histplot(data['BloodPressure'])


# In[12]:


sns.histplot(data['SkinThickness'])


# In[13]:


sns.histplot(data['Insulin'])


# In[14]:


sns.histplot(data['BMI'])


# In[15]:


data['Insulin'] = data['Insulin'].fillna(data['Insulin'].median())


# In[16]:


mean_cols = ['Glucose', 'BloodPressure', 'SkinThickness', 'BMI']
data[mean_cols] = data[mean_cols].fillna(data[mean_cols].mean())


# In[17]:


data.isnull().sum()


# In[18]:


data.dtypes.value_counts().plot(kind='bar')
```

```
# In[19]:


data['Outcome'].value_counts().plot(kind='bar')
data['Outcome'].value_counts()


# # SMOTE

# In[20]:


x_res = data.drop('Outcome', axis=1)
y_res = data['Outcome']


# In[21]:


print(x_res.shape)
print(y_res.shape)


# In[22]:


get_ipython().system('pip install imbalanced-learn')


# In[23]:


from imblearn.over_sampling import SMOTE


# In[24]:


my_smote = SMOTE(random_state=1)


# In[25]:


x, y = my_smote.fit_resample(x_res,y_res)


# In[26]:


print(x.shape)
print(y.shape)


# In[27]:


y.value_counts()
```

```
# In[28]:


rs_data = pd.concat([x, y], axis=1)
rs_data


# In[29]:


sns.pairplot(data=rs_data, hue='Outcome')


# In[30]:


plt.figure(figsize=(12,8))
sns.heatmap(rs_data.corr(), annot=True)


# In[31]:


from sklearn.model_selection import train_test_split, KFold, GridSearchCV
from sklearn.metrics import accuracy_score, average_precision_score,
f1_score, confusion_matrix, classification_report, auc, roc_curve,
roc_auc_score, precision_recall_curve


# In[32]:


x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.15,
random_state=1)


# In[33]:


x_train.shape, x_test.shape


# In[34]:


models = []
model_accuracy = []
model_f1 = []
model_auc = []


# In[35]:


from sklearn.linear_model import LogisticRegression
lr1 = LogisticRegression()
```

```python
# In[36]:


lr1.fit(x_train,y_train)


# In[37]:


lr1.score(x_train,y_train)


# In[38]:


lr1.score(x_test, y_test)


# In[39]:


from sklearn.model_selection import GridSearchCV, cross_val_score


# In[40]:


parameters = {'C':np.logspace(-5, 5, 50)}


# In[41]:


gs_lr = GridSearchCV(lr1, param_grid = parameters, cv=5, verbose=0)
gs_lr.fit(x, y)


# In[42]:


gs_lr.best_params_


# In[43]:


gs_lr.best_score_


# In[44]:


lr2 = LogisticRegression(C=0.018420699693267165, max_iter=300)


# In[46]:
```

```
lr2.fit(x_train,y_train)


# In[47]:


lr2.score(x_train,y_train)


# In[48]:


lr2.score(x_test, y_test)


# In[49]:


probs = lr2.predict_proba(x_test)                # predict probabilities
probs = probs[:, 1]                              # keep probabilities for
the positive outcome only

auc_lr = roc_auc_score(y_test, probs)            # calculate AUC
print('AUC: %.3f' %auc_lr)
fpr, tpr, thresholds = roc_curve(y_test, probs)  # calculate roc curve
plt.plot([0, 1], [0, 1], linestyle='--')         # plot no skill
plt.plot(fpr, tpr, marker='.')                   # plot the roc curve for
the model
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC (Receiver Operating Characteristics) Curve")


# In[64]:


models.append('LR')
model_accuracy.append(accuracy_score(y_test, pred_y_test))
model_f1.append(f1)
model_auc.append(auc_lr)


# In[50]:


from sklearn.tree import DecisionTreeClassifier
dt1 = DecisionTreeClassifier(random_state=0)


# In[51]:


dt1.fit(x_train,y_train)


# In[52]:
```

```python
dt1.score(x_train,y_train)


# In[53]:


dt1.score(x_test, y_test)


# In[54]:


parameters = {
    'max_depth':[1,2,3,4,5,None]}


# In[55]:


gs_dt = GridSearchCV(dt1, param_grid = parameters, cv=5, verbose=0)
gs_dt.fit(x, y)


# In[56]:


gs_dt.best_params_


# In[57]:


gs_dt.best_score_


# In[58]:


dt2 = DecisionTreeClassifier(max_depth=4)


# In[59]:


dt2.fit(x_train,y_train)


# In[60]:


dt2.score(x_train,y_train)


# In[61]:
```

```
dt2.score(x_test, y_test)


# In[62]:


probs = dt2.predict_proba(x_test)
probs = probs[:, 1]

auc_dt = roc_auc_score(y_test, probs)
print('AUC: %.3f' %auc_dt)
fpr, tpr, thresholds = roc_curve(y_test, probs)
plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr, tpr, marker='.')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC (Receiver Operating Characteristics) Curve")


# In[63]:


pred_y_test = dt2.predict(x_test)
precision, recall, thresholds = precision_recall_curve(y_test, probs)
f1 = f1_score(y_test, pred_y_test)
auc_dt_pr = auc(recall, precision)
ap = average_precision_score(y_test, probs)
print('f1=%.3f auc_pr=%.3f ap=%.3f' % (f1, auc_dt_pr, ap))
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
plt.plot(recall, precision, marker='.')
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title("Precision-Recall Curve");


# In[65]:


models.append('DT')
model_accuracy.append(accuracy_score(y_test, pred_y_test))
model_f1.append(f1)
model_auc.append(auc_dt)


# In[66]:


from sklearn.ensemble import RandomForestClassifier
rf1 = RandomForestClassifier()


# In[67]:


rf1 = RandomForestClassifier(random_state=0)


# In[68]:
```

```
rf1.fit(x_train, y_train)


# In[69]:


rf1.score(x_train, y_train)


# In[71]:


rf1.score(x_test, y_test)


# In[72]:


parameters = {
    'n_estimators': [50,100,150],
    'max_depth': [None,1,3,5,7],
    'min_samples_leaf': [1,3,5]
}


# In[73]:


gs_dt = GridSearchCV(estimator=rf1, param_grid=parameters, cv=5,
verbose=0)
gs_dt.fit(x, y)


# In[74]:


gs_dt.best_params_


# In[75]:


gs_dt.best_score_


# In[76]:


rf2 = RandomForestClassifier(max_depth=None, min_samples_leaf=1,
n_estimators=100)


# In[77]:


rf2.fit(x_train,y_train)
```

```
# In[78]:


rf2.score(x_train,y_train)


# In[79]:


rf2.score(x_test, y_test)


# In[80]:


probs = rf2.predict_proba(x_test)
probs = probs[:, 1]

auc_rf = roc_auc_score(y_test, probs)
print('AUC: %.3f' %auc_rf)
fpr, tpr, thresholds = roc_curve(y_test, probs)
plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr, tpr, marker='.')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC (Receiver Operating Characteristics) Curve")


# In[82]:


pred_y_test = rf2.predict(x_test)
precision, recall, thresholds = precision_recall_curve(y_test, probs)
f1 = f1_score(y_test, pred_y_test)
auc_rf_pr = auc(recall, precision)
ap = average_precision_score(y_test, probs)
print('f1=%.3f auc_pr=%.3f ap=%.3f' % (f1, auc_rf_pr, ap))
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
plt.plot(recall, precision, marker='.')
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title("Precision-Recall Curve")


# In[83]:


models.append('RF')
model_accuracy.append(accuracy_score(y_test, pred_y_test))
model_f1.append(f1)
model_auc.append(auc_dt)


# In[84]:
```

```
from sklearn.neighbors import KNeighborsClassifier
knn1 = KNeighborsClassifier(n_neighbors=3)


# In[85]:


knn1.fit(x_train, y_train)


# In[86]:


knn1.score(x_train,y_train)


# In[87]:


knn1.score(x_test,y_test)


# In[88]:


knn_neighbors = [i for i in range(2,16)]
parameters = {
    'n_neighbors': knn_neighbors
}


# In[89]:


gs_knn = GridSearchCV(estimator=knn1, param_grid=parameters, cv=5,
verbose=0)
gs_knn.fit(x, y)


# In[90]:


gs_knn.best_params_


# In[91]:


gs_knn.best_score_


# In[92]:


knn2 = KNeighborsClassifier(n_neighbors=3)


# In[93]:
```

```
knn2.fit(x_train, y_train)


# In[94]:


knn2.score(x_train,y_train)


# In[95]:


knn2.score(x_test,y_test)


# In[97]:


probs = knn2.predict_proba(x_test)
probs = probs[:, 1]

auc_knn = roc_auc_score(y_test, probs)
print('AUC: %.3f' %auc_knn)
fpr, tpr, thresholds = roc_curve(y_test, probs)
plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr, tpr, marker='.')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC (Receiver Operating Characteristics) Curve")


# In[98]:


pred_y_test = knn2.predict(x_test)
precision, recall, thresholds = precision_recall_curve(y_test, probs)
f1 = f1_score(y_test, pred_y_test)
auc_knn_pr = auc(recall, precision)
ap = average_precision_score(y_test, probs)
print('f1=%.3f auc_pr=%.3f ap=%.3f' % (f1, auc_knn_pr, ap))
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
plt.plot(recall, precision, marker='.')
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title("Precision-Recall Curve")


# In[99]:


models.append('KNN')
model_accuracy.append(accuracy_score(y_test, pred_y_test))
model_f1.append(f1)
model_auc.append(auc_knn)
```

```
# In[100]:


from sklearn.svm import SVC
svm1 = SVC(kernel='rbf')


# In[102]:


svm1.fit(x_train, y_train)


# In[104]:


svm1.score(x_train, y_train)


# In[105]:


svm1.score(x_test, y_test)


# In[106]:


parameters = {
    'C':[1, 5, 10, 15, 20, 25],
    'gamma':[0.001, 0.005, 0.0001, 0.00001]
}


# In[107]:


gs_svm = GridSearchCV(estimator=svm1, param_grid=parameters, cv=5,
verbose=0)
gs_svm.fit(x, y)


# In[108]:


gs_svm.best_params_


# In[109]:


gs_svm.best_score_


# In[110]:


svm2 = SVC(kernel='rbf', C=20, gamma=0.005, probability=True)
```

```
# In[111]:


svm2.fit(x_train, y_train)


# In[112]:


svm2.score(x_train, y_train)


# In[113]:


svm2.score(x_test, y_test)


# In[115]:


probs = svm2.predict_proba(x_test)
probs = probs[:, 1]

auc_svm = roc_auc_score(y_test, probs)
print('AUC: %.3f' %auc_svm)
fpr, tpr, thresholds = roc_curve(y_test, probs)
plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr, tpr, marker='.')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC (Receiver Operating Characteristics) Curve")


# In[116]:


pred_y_test = svm2.predict(x_test)
precision, recall, thresholds = precision_recall_curve(y_test, probs)
f1 = f1_score(y_test, pred_y_test)
auc_svm_pr = auc(recall, precision)
ap = average_precision_score(y_test, probs)
print('f1=%.3f auc_pr=%.3f ap=%.3f' % (f1, auc_svm_pr, ap))
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
plt.plot(recall, precision, marker='.')
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title("Precision-Recall Curve")


# In[117]:


models.append('SVM')
model_accuracy.append(accuracy_score(y_test, pred_y_test))
model_f1.append(f1)
```

```
model_auc.append(auc_svm)


# In[118]:


from sklearn.ensemble import AdaBoostClassifier
ada1 = AdaBoostClassifier(n_estimators=100)


# In[119]:


ada1.fit(x_train,y_train)


# In[120]:


ada1.score(x_train,y_train)


# In[121]:


ada1.score(x_test, y_test)


# In[122]:


parameters = {'n_estimators': [100,200,300,400,500,700,1000]}


# In[123]:


gs_ada = GridSearchCV(ada1, param_grid = parameters, cv=5, verbose=0)
gs_ada.fit(x, y)


# In[124]:


gs_ada.best_params_


# In[125]:


gs_ada.best_score_


# In[126]:


ada2 = AdaBoostClassifier(n_estimators=500)
ada2.fit(x_train,y_train)
```

```
# In[127]:


ada2.score(x_train,y_train)


# In[128]:


ada2.score(x_test, y_test)


# In[129]:


probs = ada2.predict_proba(x_test)
probs = probs[:, 1]

auc_ada = roc_auc_score(y_test, probs)
print('AUC: %.3f' %auc_ada)
fpr, tpr, thresholds = roc_curve(y_test, probs)
plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr, tpr, marker='.')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC (Receiver Operating Characteristics) Curve")


# In[130]:


pred_y_test = ada2.predict(x_test)
precision, recall, thresholds = precision_recall_curve(y_test, probs)
f1 = f1_score(y_test, pred_y_test)
auc_ada_pr = auc(recall, precision)
ap = average_precision_score(y_test, probs)
print('f1=%.3f auc_pr=%.3f ap=%.3f' % (f1, auc_ada_pr, ap))
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
plt.plot(recall, precision, marker='.')
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title("Precision-Recall Curve")


# In[131]:


models.append('ADA')
model_accuracy.append(accuracy_score(y_test, pred_y_test))
model_f1.append(f1)
model_auc.append(auc_ada)


# In[132]:
```

```
from xgboost import XGBClassifier
xgb1 = XGBClassifier(use_label_encoder=False, objective =
'binary:logistic', nthread=4, seed=10)


# In[133]:


xgb1.fit(x_train, y_train)


# In[134]:


xgb1.score(x_train, y_train)


# In[135]:


xgb1.score(x_test, y_test)


# In[136]:


parameters = {
    'max_depth': range (2, 10, 1),
    'n_estimators': range(60, 220, 40),
    'learning_rate': [0.1, 0.01, 0.05]
}


# In[137]:


gs_xgb = GridSearchCV(xgb1, param_grid = parameters, scoring = 'roc_auc',
n_jobs = 10, cv=5, verbose=0)
gs_xgb.fit(x, y)


# In[138]:


gs_xgb.best_params_


# In[139]:


gs_xgb.best_score_


# In[140]:


xgb2 = XGBClassifier(use_label_encoder=False, objective =
'binary:logistic',
```

```
                    nthread=4, seed=10, learning_rate= 0.05, max_depth=
7, n_estimators= 180)


# In[141]:


xgb2.fit(x_train,y_train)


# In[142]:


xgb2.score(x_train,y_train)


# In[143]:


xgb2.score(x_test, y_test)


# In[145]:


probs = xgb2.predict_proba(x_test)
probs = probs[:, 1]

auc_xgb = roc_auc_score(y_test, probs)
print('AUC: %.3f' %auc_xgb)
fpr, tpr, thresholds = roc_curve(y_test, probs)
plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr, tpr, marker='.')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC (Receiver Operating Characteristics) Curve")


# In[146]:


pred_y_test = xgb2.predict(x_test)
precision, recall, thresholds = precision_recall_curve(y_test, probs)
f1 = f1_score(y_test, pred_y_test)
auc_xgb_pr = auc(recall, precision)
ap = average_precision_score(y_test, probs)
print('f1=%.3f auc_pr=%.3f ap=%.3f' % (f1, auc_xgb_pr, ap))
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
plt.plot(recall, precision, marker='.')
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title("Precision-Recall Curve")


# In[147]:


models.append('XGB')
```

```
model_accuracy.append(accuracy_score(y_test, pred_y_test))
model_f1.append(f1)
model_auc.append(auc_xgb)


# In[148]:


model_summary =
pd.DataFrame(zip(models,model_accuracy,model_f1,model_auc), columns =
['model','accuracy','f1_score','auc'])
model_summary = model_summary.set_index('model')


# In[149]:


model_summary.plot(figsize=(16,5))
plt.title("Comparison of Different Classification Algorithms")


# In[150]:


model_summary


# In[151]:


main_model = xgb2


# In[154]:


cls_report = classification_report(y_test, main_model.predict(x_test))
print(cls_report)


# In[155]:


confusion = confusion_matrix(y_test, main_model.predict(x_test))


# In[156]:


confusion


# In[157]:


TP = confusion[1,1]
TN = confusion[0,0]
FP = confusion[0,1]
```

```
FN = confusion[1,0]

Accuracy = (TP+TN)/(TP+TN+FP+FN)
Precision = TP/(TP+FP)
Sensitivity = TP/(TP+FN)
Specificity = TN/(TN+FP)


# In[158]:


print("Accuracy: %.3f"%Accuracy)
print("Precision: %.3f"%Precision)
print("Sensitivity: %.3f"%Sensitivity)
print("Specificity: %.3f"%Specificity)
print("AUC: %.3f"%auc_rf)


# In[ ]:
```