Python Code Compilation

Program 1: K-Means Clustering

```
import pandas as pd
import numpy as np
import random as rd
import matplotlib.pyplot as plt

data = pd.read_csv('clustering.csv')
data.head()
X = data[["LoanAmount","ApplicantIncome"]]
plt.scatter(X["ApplicantIncome"],X["LoanAmount"],c='black')
plt.xlabel('AnnualIncome')
plt.ylabel('Loan Amount (In Thousands)')
plt.show()
K = 3
Centroids = X.sample(n=K)
diff = 1
j = 0

while diff != 0:
    XD = X.copy(deep=True)
    i = 1

    for index1, row_c in Centroids.iterrows():
        ED = []
        for index2, row_d in XD.iterrows():
            d1 = (row_c["ApplicantIncome"] - row_d["ApplicantIncome"]) ** 2
            d2 = (row_c["LoanAmount"] - row_d["LoanAmount"]) ** 2
            d = np.sqrt(d1 + d2)
            ED.append(d)
        X.loc[:, i] = ED
        i += 1

    C = []
    for index, row in X.iterrows():
        min_dist = row[1]
        pos = 1
        for i in range(K):
            if row[i + 1] < min_dist:
                min_dist = row[i + 1]
                pos = i + 1
        C.append(pos)
    X.loc[:, "Cluster"] = C

    Centroids_new = X.groupby(["Cluster"]).mean()[["LoanAmount", "ApplicantIncome"]]

    if j == 0:
        diff = 1
        j += 1
    else:
        diff = (Centroids_new["LoanAmount"] - Centroids["LoanAmount"]).sum() +
(Centroids_new["ApplicantIncome"] - Centroids["ApplicantIncome"]).sum()
        print(diff)
    Centroids = Centroids_new
color=['blue','green','cyan']
for k in range(K):
    data=X[X["Cluster"]==k+1]
    plt.scatter(data["ApplicantIncome"],data["LoanAmount"],c=color[k])
plt.scatter(Centroids["ApplicantIncome"],Centroids["LoanAmount"],c='red')
plt.xlabel('Income')
plt.ylabel('Loan Amount (In Thousands)')
plt.show()
```

Program 2: Polynomial Regression

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

# Importing the dataset
datas = pd.read_csv('data.csv')
datas
X = datas.iloc[:, 1:2].values
y = datas.iloc[:, 2].values
# Features and the target variables
X = datas.iloc[:, 1:2].values
y = datas.iloc[:, 2].values

# Fitting Linear Regression to the dataset
from sklearn.linear_model import LinearRegression
lin = LinearRegression()

lin.fit(X, y)

from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=4)
X_poly = poly.fit_transform(X)

poly.fit(X_poly, y)
lin2 = LinearRegression()
lin2.fit(X_poly, y)

plt.scatter(X, y, color='blue')

plt.plot(X, lin.predict(X), color='red')
plt.title('Linear Regression')
plt.xlabel('Temperature')
plt.ylabel('Pressure')

plt.show()

plt.scatter(X, y, color='blue')

plt.plot(X, lin2.predict(poly.fit_transform(X)),
    color='red')
```

```
plt.title('Polynomial Regression')
plt.xlabel('Temperature')
plt.ylabel('Pressure')

plt.show()

pred = 110.0
predarray = np.array([[pred]])
lin.predict(predarray)

pred2 = 110.0
pred2array = np.array([[pred2]])
lin2.predict(poly.fit_transform(pred2array))
```

Program 3 : SVM

```
from sklearn.datasets import load_breast_cancer
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.svm import SVC
import matplotlib.pyplot as plt


cancer = load_breast_cancer()
X = cancer.data[:, :2]
y = cancer.target


svm = SVC(kernel = "rbf", gamma = 0.5, C = 1.0)
svm.fit(X, y)
DecisionBoundaryDisplay.from_estimator(svm,
                    X,
                    response_method = "predict",
                    cmap = plt.cm.Spectral,
                    alpha = 0.8,
                    xlabel = cancer.feature_names[0],
                    ylabel = cancer.feature_names[1])
plt.scatter(X[:,0], X[:,1], c = y, s = 20, edgecolors = "k")
plt.show()
```

Program 4 : Random forest

```
from sklearn import datasets
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import pandas as pd

iris = datasets.load_iris()

print(iris.target_names)
print(iris.feature_names)

X, y = datasets.load_iris( return_X_y = True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30)

data = pd.DataFrame({'sepallength': iris.data[:, 0], 'sepalwidth': iris.data[:, 1], 'petallength': iris.data[:, 2], 'petalwidth': iris.data[:, 3],
                     'species': iris.target})
print(data.head())

clf = RandomForestClassifier(n_estimators = 100)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print()
print("ACCURACY OF THE MODEL:", metrics.accuracy_score(y_test, y_pred))
```

```
clf.predict([[3, 3, 2, 2]])

feature_imp = pd.Series(clf.feature_importances_, index =
iris.feature_names).sort_values(ascending = False)

feature_imp
```