(1) For the dataset, I picked up a dataset from kaggle but that was small as it gave less r2 score so I have enlarged the dataset and made it sufficiently large

(2) I used the kaggle dataset to produce new augmented dataset, encoded string data, standardised the data, tackled null values and then applied various regression techniques and neural network to calculate which one is best in terms of accuracy (r2 score).

The roadblocks I faced during the task is that the accuracy decreased for a much larger dataset and even for a smaller dataset. I had to find the a length of dataset which gave maximum accuracy. Also I faced issues in enlarging the dataset as I had a choose a random values which had to be added to age and height (some seemed large and some small). Also I had to select the hyperparameters especially in neural network which gave best results.

(3) To run the code open the code.ipynb file and simply run all the cells

(4) Data encoding -

```
label_encoder = LabelEncoder()
df['Gender'] = label_encoder.fit_transform(df['Gender'])
df['Smoke'] = label_encoder.fit_transform(df['Smoke'])
df['Caesarean'] = label_encoder.fit_transform(df['Caesarean'])
```

Handling null values -

```
df.isnull().sum()
```

```
LungCap(cc)       20
Age( years)       20
Height(inches)    14
Smoke              0
Gender             0
Caesarean          0
No of children     0
Weight (kg)       14
dtype: int64
```

```
df = df.fillna(df.median())
```

```
df.isnull().sum()
```

```
LungCap(cc)       0
Age( years)       0
Height(inches)    0
Smoke             0
Gender            0
Caesarean         0
No of children    0
Weight (kg)       0
dtype: int64
```

To enlarge the dataset -

```python
for i in range(1):
    to_add_data=pd.DataFrame()
    to_add_data['LungCap(cc)']=None
    to_add_data['Age( years)']=None
    to_add_data['Height(inches)']=None
    to_add_data['Smoke']=None
    to_add_data['Gender']=None
    to_add_data['Caesarean']=None
    to_add_data['No of children']=None
    to_add_data['Weight (kg)']=None
    to_add_data['LungCap(cc)'] =df['LungCap(cc)']
    to_add_data['Age( years)'] =df['Age( years)'] + np.random.randint(-5, 5)
    to_add_data['Height(inches)'] =df['Height(inches)'] + np.random.uniform(-5, 5)
    to_add_data['Smoke']=np.random.choice([0,1,2,3])
    to_add_data['Gender']=np.random.choice([0,1])
    to_add_data['Caesarean']=np.random.choice([0,1,2,3])
    to_add_data['No of children']=np.random.choice([0,1,2,3,4])
    to_add_data['Weight (kg)']=df['Weight (kg)'] + np.random.uniform(-10, 10)
    df = pd.concat([df, to_add_data], ignore_index=True)
```

Using various prediction models -

```python
model = LinearRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
r2 = r2_score(y_test, predictions)
print(r2)
```

```
0.8080915193526845
```

```python
model = DecisionTreeRegressor()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
r2 = r2_score(y_test, predictions)
print(r2)
```

```
0.6740964238449471
```

```python
model = RandomForestRegressor()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
r2 = r2_score(y_test, predictions)
print(r2)
```

```
0.7938138168059157
```

```python
model = SVR()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
r2 = r2_score(y_test, predictions)
print(r2)
```

```
0.8028043121418489
```

```python
model = GradientBoostingRegressor()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
r2 = r2_score(y_test, predictions)
print(r2)
```

```
0.822371274222428
```

```python
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1)
])
model.compile(optimizer='adam', loss='mean_squared_error')
history = model.fit(X_train, y_train, epochs=500, batch_size=16, validation_split=0.2, verbose=0)
y_pred = model.predict(X_test)
r2 = r2_score(y_test, y_pred)
print(r2)
```

```
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass a
n `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in th
e model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 924us/step
0.8256655097796408
```