1. Breadth First Search (BFS) on a Tree

```
def bfs(tree, start, goal):
    queue = [start]
    while queue:
        node = queue.pop(0)
        if node == goal:
            return f"Goal {goal} found"
        queue.extend(tree.get(node, []))
    return "Goal not found"

tree = {2: [7, 5], 7: [11], 5: [], 11: []}
print(bfs(tree, 2, 11))
```

2. Breadth First Search (BFS) on a Graph

```
def bfs(graph, root):
    visited, queue = set(), [root]
    while queue:
        vertex = queue.pop(0)
        if vertex not in visited:
            visited.add(vertex)
            queue.extend(set(graph[vertex]) - visited)
    return visited

graph = {5: [7, 8], 7: [3, 4], 8: [], 3: [], 4: []}
print(bfs(graph, 5))
```

3. Depth First Search (DFS) on a Graph

```
def dfs(graph, start, goal, visited=None):
    if visited is None:
        visited = set()
    visited.add(start)
    if start == goal:
        return True
    for neighbor in graph.get(start, []):
        if neighbor not in visited and dfs(graph, neighbor, goal, visited):
            return True
    return False

graph = {'S': ['A', 'K'], 'A': ['B', 'C'], 'K': ['I', 'J'], 'C': ['H'], 'H': ['M'], 'M':
['N'], 'N': ['O']}
print(dfs(graph, 'S', 'O'))
```

4. Depth First Search (DFS) for a Goal Node

```
graph = {1: [2, 3], 2: [4, 5], 3: [6], 4: [], 5: [], 6: []}
def dfs(node, goal, graph, visited=set()):
    if node == goal:
        return f"Goal {goal} found"
    visited.add(node)
    for neighbor in graph.get(node, []):
        if neighbor not in visited:
            result = dfs(neighbor, goal, graph, visited)
```

```
            if result:
                return result
    return "Goal not found"

print(dfs(1, 6, graph))
```

5. Best First Search on a Tree

```
from queue import PriorityQueue

def best_first_search(tree, start, goal):
    pq = PriorityQueue()
    pq.put((0, start))
    while not pq.empty():
        cost, node = pq.get()
        if node == goal:
            return f"Goal {goal} found"
        for child in tree.get(node, []):
            pq.put((child[1], child[0]))
    return "Goal not found"

tree = {10: [(15, 2), (30, 3)], 15: [(100, 4)], 30: []}
print(best_first_search(tree, 10, 100))
```

6. Best First Search on a Graph

```
from queue import PriorityQueue
```

```
def best_first_search(graph, start, goal):
    pq = PriorityQueue()
    pq.put((0, start))
    visited = set()
    while not pq.empty():
        _, current = pq.get()
        if current in visited:
            continue
        visited.add(current)
        if current == goal:
            return f"Goal {goal} found"
        for neighbor, cost in graph.get(current, []):
            pq.put((cost, neighbor))
    return "Goal not found"

graph = {'S': [('A', 1), ('B', 4)], 'A': [('G', 2)], 'B': [('G', 1)], 'G': []}
print(best_first_search(graph, 'S', 'G'))
```

7. A* Algorithm on a Graph

```
from queue import PriorityQueue

def a_star(graph, start, goal, heuristic):
    pq = PriorityQueue()
    pq.put((0, start, 0))
    visited = {}
    while not pq.empty():
```

```
        f_cost, current, g_cost = pq.get()
        if current == goal:
            return f"Goal {goal} reached with cost {g_cost}"
        if current in visited:
            continue
        visited[current] = g_cost
        for neighbor, cost in graph.get(current, []):
            new_g = g_cost + cost
            f_cost = new_g + heuristic[neighbor]
            pq.put((f_cost, neighbor, new_g))
    return "Goal not found"

graph = {'S': [('A', 1), ('B', 4)], 'A': [('G', 2)], 'B': [('G', 1)], 'G': []}
heuristic = {'S': 5, 'A': 3, 'B': 4, 'G': 0}
print(a_star(graph, 'S', 'G', heuristic))
```

8. Credit Card Fraud Detection

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

data = pd.read_csv('credit_card_data.csv')  # Example placeholder
X = data.drop('fraud', axis=1)
y = data['fraud']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
model = RandomForestClassifier()

model.fit(X_train, y_train)

predictions = model.predict(X_test)

print(f"Accuracy: {accuracy_score(y_test, predictions)}")
```

9. E-mail Spam Detection

```
from sklearn.feature_extraction.text import CountVectorizer

from sklearn.naive_bayes import MultinomialNB

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score

import pandas as pd


data = pd.read_csv('spam_dataset.csv')  # Example placeholder

X = data['message']

y = data['label']

vectorizer = CountVectorizer()

X = vectorizer.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

model = MultinomialNB()

model.fit(X_train, y_train)

predictions = model.predict(X_test)

print(f"Accuracy: {accuracy_score(y_test, predictions)}")
```

10. Lung Cancer Detection

```
from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import accuracy_score
import pandas as pd

data = pd.read_csv('lung_cancer_data.csv')  # Example placeholder
X = data.drop('cancer', axis=1)
y = data['cancer']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = RandomForestClassifier()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
print(f"Accuracy: {accuracy_score(y_test, predictions)}")
```

11. EDA for Gold Price Prediction

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

data = pd.read_csv('gold_price.csv')  # Example placeholder
print(data.describe())
sns.heatmap(data.corr(), annot=True)
plt.show()
```

12. NLP Emotion Detection Dataset

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

```
import pandas as pd


data = pd.read_csv('emotion_dataset.csv')  # Example placeholder
X = data['text']
y = data['emotion']
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = LogisticRegression()
model.fit(X_train, y_train)
print(f"Accuracy: {model.score(X_test, y_test)}")
```

13. NLP for Twitter Dataset

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
import pandas as pd


data = pd.read_csv('twitter_data.csv')  # Example placeholder
X = data['tweet']
y = data['sentiment']
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = MultinomialNB()
model.fit(X_train, y_train)
```

```
print(f"Accuracy: {model.score(X_test, y_test)}")
```

14. NLP for Fake News Dataset

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import pandas as pd

data = pd.read_csv('fake_news.csv')  # Example placeholder
X = data['news']
y = data['label']
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = RandomForestClassifier()
model.fit(X_train, y_train)
print(f"Accuracy: {model.score(X_test, y_test)}")
```

15. Analyze and Load Flight Price Dataset

```
import pandas as pd

data = pd.read_csv('flight_prices.csv')  # Example placeholder
X = data.drop('price', axis=1)
y = data['price']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
print(X_train.head(), y_train.head())
```

16. Analyze and Load Iris Dataset

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

iris = load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
print("Training and Testing Split Successful!")
```

17. Simple Linear Regression for House Prices

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import pandas as pd

data = pd.read_csv('house_prices.csv')  # Example placeholder
X = data[['size']]  # Example feature
y = data['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = LinearRegression()
model.fit(X_train, y_train)
print(f"Model Coefficients: {model.coef_}, Intercept: {model.intercept_}")
```

18. Simple Linear Regression for Stock Prices

```
from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split

import pandas as pd


data = pd.read_csv('stock_prices.csv')  # Example placeholder

X = data[['date']]  # Example feature

y = data['price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

model = LinearRegression()

model.fit(X_train, y_train)

print(f"Model Coefficients: {model.coef_}, Intercept: {model.intercept_}")
```

19. Multiple Linear Regression for Car Prices

```
from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split

import pandas as pd


data = pd.read_csv('car_prices.csv')  # Example placeholder

X = data[['age', 'mileage']]  # Example features

y = data['price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

model = LinearRegression()

model.fit(X_train, y_train)

print(f"Model Coefficients: {model.coef_}, Intercept: {model.intercept_}")
```

20. Multiple Linear Regression for Wine Quality

```
from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split

import pandas as pd


data = pd.read_csv('wine_quality.csv')  # Example placeholder

X = data[['alcohol', 'sulphates']]  # Example features

y = data['quality']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

model = LinearRegression()

model.fit(X_train, y_train)

print(f"Model Coefficients: {model.coef_}, Intercept: {model.intercept_}")
```