

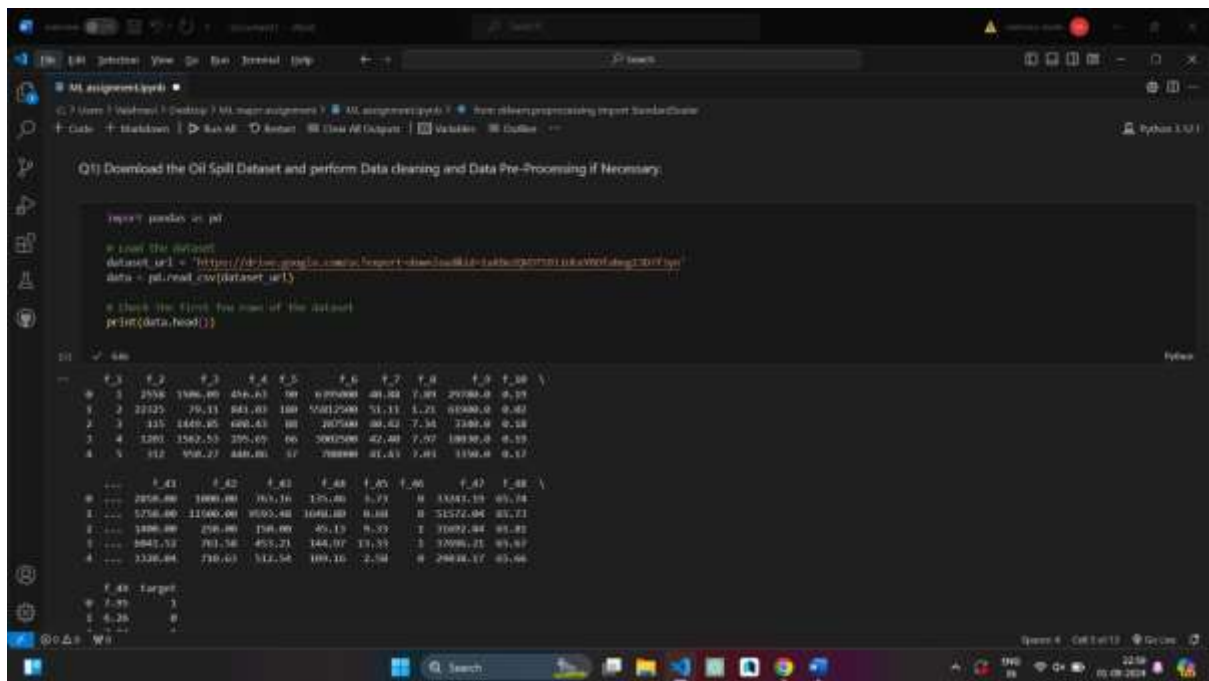
Name: Vaishnavi Digambar Jevale

Email id: vaishnavijevale59@gmail.com

Phone no.: 9021573607

Q1) Download the Oil Spill Dataset and perform Data cleaning and Data Pre-Processing if Necessary.

INPUT:



The screenshot shows a Jupyter Notebook interface with the following code and output:

```
import pandas as pd

# Load the dataset
dataset_url = "https://raw.githubusercontent.com/leventkoc/OilSpillDataset/master/oilspill_dataset.csv"
data = pd.read_csv(dataset_url)

# Check the first five rows of the dataset
print(data.head())
```

The output displays the first five rows of the dataset, which are structured as follows:

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_10
0	1	2558	1586.00	456.83	00	0.000000	40.04	7.39	20780.0	0.19
1	2	22125	79.11	841.03	100	0.012500	51.11	1.21	60900.0	0.02
2	3	115	1440.05	680.43	00	0.075000	80.42	7.34	7100.0	0.18
3	4	1203	1582.53	370.09	00	0.007500	42.40	7.07	10030.0	0.19
4	5	312	950.27	440.00	07	0.000000	01.43	7.01	1150.0	0.17

Below the first five rows, there is a section for the 'f_10' and 'target' columns:

	f_10	target
0	2050.00	1000.00
1	5750.00	11000.00
2	1200.00	250.00
3	8047.53	701.14
4	1330.04	710.63

OUTPUT:

```

# Check the first five rows of the dataset
print(data.head())

# Check the full column
print(data.info())

```

Output of `print(data.head())`:

	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_10
0	3	2578	1546.86	456.43	80	0.995089	30.28	7.39	29780.0	0.19	
1	2	22325	79.11	641.03	380	0.992508	51.11	1.21	40980.0	0.42	
2	3	115	1449.48	686.43	80	0.97500	86.43	7.34	3180.0	0.18	
3	4	1301	1542.53	395.45	46	0.982589	82.88	7.97	18830.0	0.39	
4	5	333	950.27	440.86	37	0.98000	81.43	7.03	1390.0	0.37	

Output of `print(data.info())`:

```

# Check the full column
print(data.info())

```

Output of `print(data.info())`:

```

# Check the full column
print(data.info())

```

```

# Check the first five rows of the dataset
print(data.head())

# Check the full column
print(data.info())

```

Output of `print(data.head())`:

	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_10
0	3	2578	1546.86	456.43	80	0.995089	30.28	7.39	29780.0	0.19	
1	2	22325	79.11	641.03	380	0.992508	51.11	1.21	40980.0	0.42	
2	3	115	1449.48	686.43	80	0.97500	86.43	7.34	3180.0	0.18	
3	4	1301	1542.53	395.45	46	0.982589	82.88	7.97	18830.0	0.39	
4	5	333	950.27	440.86	37	0.98000	81.43	7.03	1390.0	0.37	

Output of `print(data.info())`:

```

# Check the full column
print(data.info())

```

Output of `print(data.info())`:

```

# Check the full column
print(data.info())

```

Q2) Use various methods such as Handling null values, One-Hot Encoding, Imputation, and Scaling of Data Pre-Processing where necessary.

INPUT:

```
from sklearn.preprocessing import StandardScaler

# Assuming there are no categorical columns (since this is a typical numerical dataset for ML tasks)
# Feature scaling
scaler = StandardScaler()
features = data.drop('target', axis=1)
target = data['target']
scaled_features = scaler.fit_transform(features)

# Combining scaled features with the target
data_scaled = pd.DataFrame(scaled_features, columns=features.columns)
data_scaled['target'] = target

print(data_scaled.head())
```

	f_1	f_2	f_3	f_4	f_5	f_6	f_7
0	-1.288972	1.152390	1.346434	-0.793007	0.129057	1.409891	-0.185871
1	-1.225524	11.389546	1.032273	-0.857362	2.116766	18.174844	0.518985
2	-1.238136	-0.333818	1.252645	-0.904901	0.807944	-0.125929	-0.223858
3	-1.194727	0.449611	1.649556	-1.101091	0.209795	0.581114	-0.906293
4	-1.173329	-0.010794	0.419520	-0.823188	-1.03352	0.062091	-0.142684

	f_8	f_9	f_10	...	f_41	f_42	f_43
0	-0.345387	3.165385	-0.163460	...	1.913873	0.000507	0.959757
1	-2.201487	7.109384	-2.226754	...	0.010555	15.485196	17.486186
2	-0.498489	-0.071549	-0.454242	...	0.405538	-0.246348	-0.137438
3	-0.328864	4.725678	0.243469	...	5.101741	0.467147	9.270349
4	-0.344864	-0.073564	-0.165484	...	0.305669	0.395881	0.443443

OUTPUT:

```
# Combining scaled features with the target
data_scaled = pd.DataFrame(scaled_features, columns=features.columns)
data_scaled['target'] = target

print(data_scaled.head())
```

	f_1	f_2	f_3	f_4	f_5	f_6	f_7
0	-1.288972	1.152390	1.346434	-0.793007	0.129057	1.409891	-0.185871
1	-1.225524	11.389546	1.032273	-0.857362	2.116766	18.174844	0.518985
2	-1.238136	-0.333818	1.252645	-0.904901	0.807944	-0.125929	-0.223858
3	-1.194727	0.449611	1.649556	-1.101091	0.209795	0.581114	-0.906293
4	-1.173329	-0.010794	0.419520	-0.823188	-1.03352	0.062091	-0.142684

	f_8	f_9	f_10	...	f_41	f_42	f_43
0	-0.345387	3.165385	-0.163460	...	1.913873	0.000507	0.959757
1	-2.201487	7.109384	-2.226754	...	0.010555	15.485196	17.486186
2	-0.498489	-0.071549	-0.454242	...	0.405538	-0.246348	-0.137438
3	-0.328864	4.725678	0.243469	...	5.101741	0.467147	9.270349
4	-0.344864	-0.073564	-0.165484	...	0.305669	0.395881	0.443443

	f_44	f_45	f_46	f_47	f_48	f_49	target
0	0.210714	-0.255448	-0.303248	3.696707	0.380736	-0.958177	1
1	11.181581	-0.010152	-0.361248	0.302101	0.307769	-0.679664	0
2	-0.445985	-0.050054	2.660278	7.400400	0.375456	-0.896212	1
3	0.186875	1.054443	2.660278	4.136762	0.302084	-0.017042	1
4	0.022483	-0.484113	-0.303248	3.677971	0.303443	-0.204751	0

[5 rows x 10 columns]

Q3) Derive some insights from the dataset.

INPUT:

```
Q3) Derive some insights from the dataset.

# Basic statistics
print(data_scaled.describe())

# Correlation matrix
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 8))
sns.heatmap(data_scaled.corr(), annot=True, cmap='magma')
plt.show()

# Class distribution
print(data_scaled['target'].value_counts())
```

Python

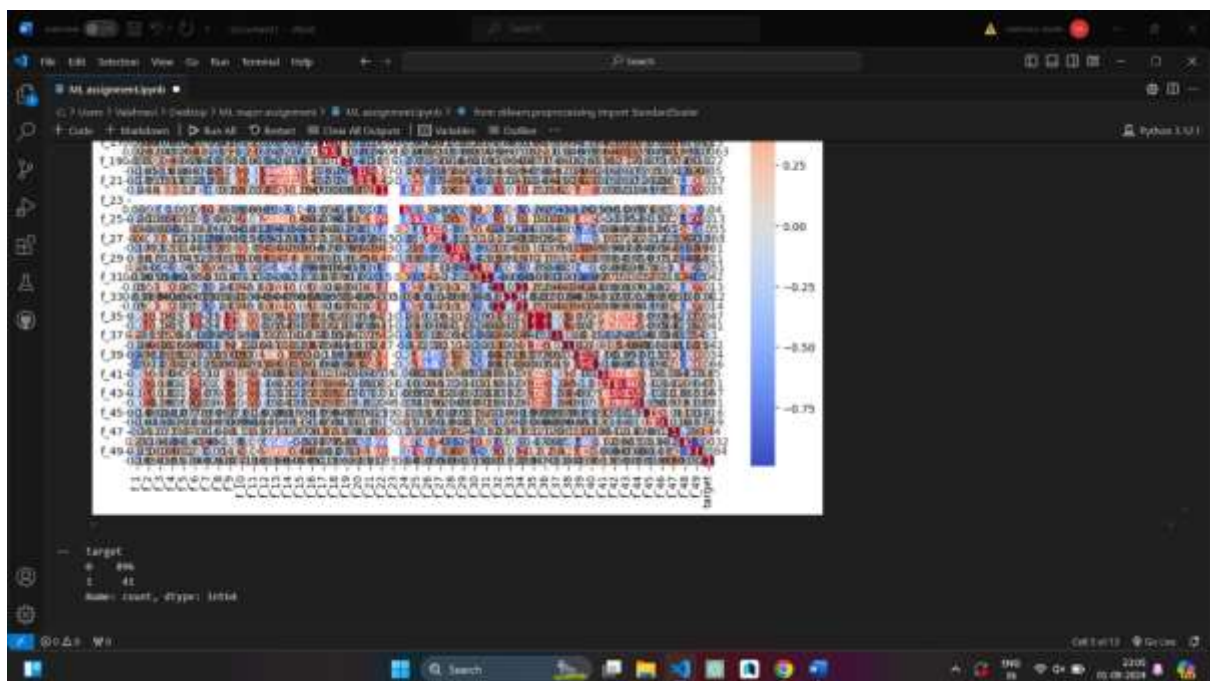
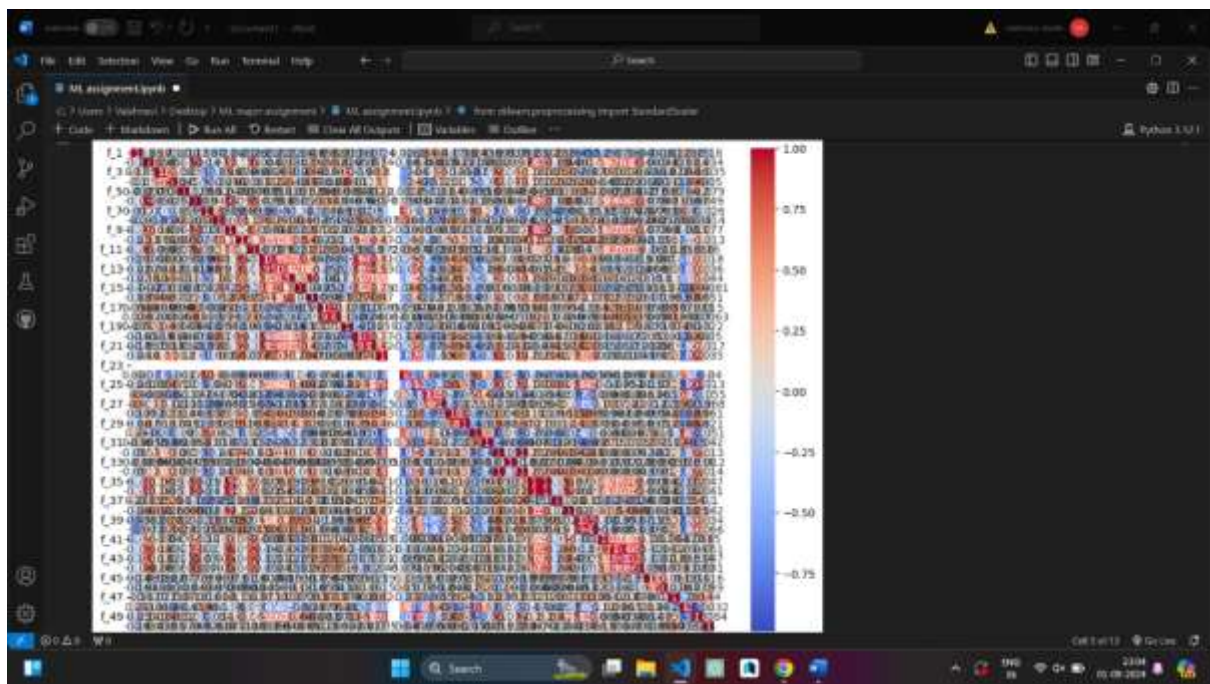
OUTPUT:

```
count    9.170000e+02    9.170000e+02    9.170000e+02    9.170000e+02    9.170000e+02
mean     -1.516615e-17    3.832676e-17    -6.466533e-17    1.213307e-16    -1.440002e-16
std       1.000534e+00    1.000534e+00    1.000534e+00    1.000534e+00    1.000534e+00
min       -1.246022e+00    -1.012018e-01    -1.101000e+00    -1.004992e+00    -1.455529e+00
25%       7.789722e-01    1.620182e-01    1.401000e+00    0.107950e-01    -9.643000e-01
50%       2.788269e-01    1.187136e-01    9.643771e-01    2.996796e-01    -2.433000e-01
75%       6.330738e-01    1.040143e-01    8.751300e-01    7.431917e-01    2.251896e-01
max       4.183866e+00    1.040143e+01    1.021700e+01    3.547300e+00    2.134760e+00

count    9.170000e+02    9.170000e+02    9.170000e+02    9.170000e+02    9.170000e+02
mean     -1.274958e-17    3.832676e-16    2.256613e-16    0.000000e+00    1.205138e-16
std       1.000534e+00    1.000534e+00    1.000534e+00    1.000534e+00    1.000534e+00
min       -1.246022e+00    -1.756015e+00    2.133327e+00    0.001000e+00    2.226756e+00
25%       1.683673e-01    7.546423e-01    6.629250e-01    0.314797e-01    0.758996e-01
50%       5.593584e-01    2.574501e-01    2.568020e-01    -0.226719e-01    -2.126777e-01
75%       1.147896e-01    2.235601e-01    4.550126e-01    -0.001931e-01    4.320456e-01
max       1.842346e+01    1.490333e+00    6.530526e+00    19.200377e+00    5.769532e+00

count    9.170000e+02    9.170000e+02    9.170000e+02    9.170000e+02
mean     -8.408886e-17    -1.116133e-17    1.316623e-17    1.013269e-17
std       1.000534e+00    1.000534e+00    1.000534e+00    1.000534e+00
min       -5.328790e-01    -5.077956e-01    -4.787296e-01    -7.828596e-01
75%      0.000000e+00
max       1.000000e+00

[6 rows x 36 columns]
Output is truncated. View as a table or open in a text editor. Adjust cell output settings...
```



Q4) Apply various Machine Learning techniques to predict the output in the target column, make use of Bagging and Ensemble as required, and find the best model by evaluating the model using Model evaluation techniques.

INPUT:

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import accuracy_score, classification_report

X = data_scaled.drop('target', axis=1)
y = data_scaled['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Random Forest
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)

print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
print(classification_report(y_test, y_pred_rf))

# Gradient Boosting
gb_model = GradientBoostingClassifier(random_state=42)
gb_model.fit(X_train, y_train)
y_pred_gb = gb_model.predict(X_test)

print("Gradient Boosting Accuracy:", accuracy_score(y_test, y_pred_gb))
print(classification_report(y_test, y_pred_gb))
```

OUTPUT:

```
# Gradient Boosting
gb_model = GradientBoostingClassifier(random_state=42)
gb_model.fit(X_train, y_train)
y_pred_gb = gb_model.predict(X_test)

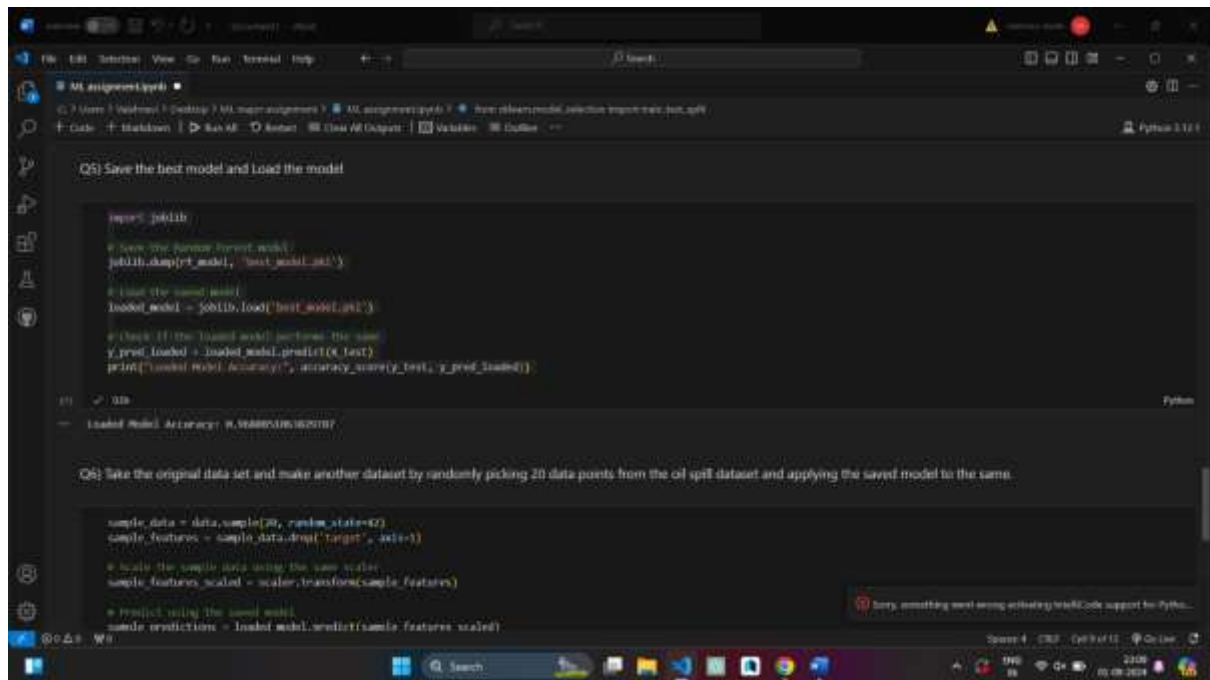
print("Gradient Boosting Accuracy:", accuracy_score(y_test, y_pred_gb))
print(classification_report(y_test, y_pred_gb))
```

	precision	recall	f1-score	support
0	0.98	0.99	0.98	102
1	0.58	0.33	0.43	6
accuracy			0.67	108
macro avg	0.78	0.66	0.69	108
weighted avg	0.86	0.93	0.95	108

	precision	recall	f1-score	support
0	0.99	0.98	0.98	102
1	0.62	0.33	0.47	6
accuracy			0.98	108
macro avg	0.81	0.61	0.70	108
weighted avg	0.98	0.98	0.98	108

Q5) Save the best model and Load the model

INPUT:



```
import joblib

# Save the RandomForest model
joblib.dump(rf_model, "best_model.pkl")

# Load the saved model
loaded_model = joblib.load("best_model.pkl")

# Check if the loaded model performs the same
y_pred_loaded = loaded_model.predict(x_test)
print("Loaded Model Accuracy:", accuracy_score(y_test, y_pred_loaded))
```

Loaded Model Accuracy: 0.9988051063020187

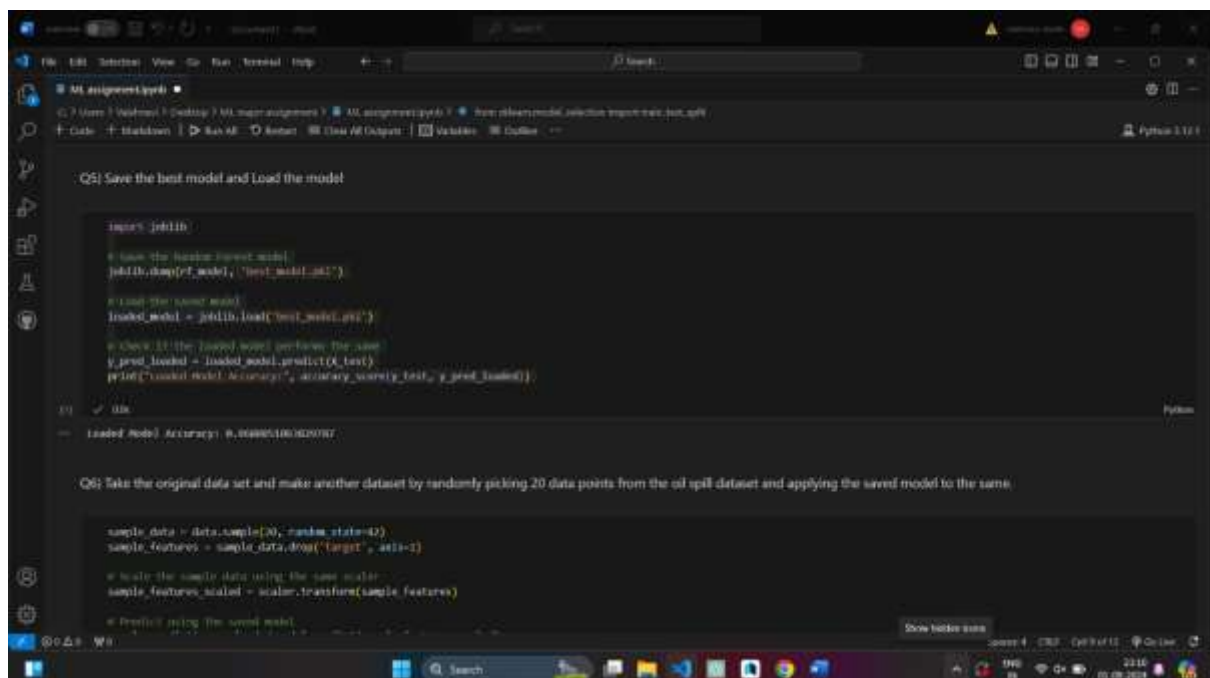
Q6) Take the original data set and make another dataset by randomly picking 20 data points from the oil spill dataset and applying the saved model to the same.

```
sample_data = data.sample(20, random_state=42)
sample_features = sample_data.drop("target", axis=1)

# Scale the sample data using the same scaler
sample_features_scaled = scaler.transform(sample_features)

# Predict using the saved model
sample_predictions = loaded_model.predict(sample_features_scaled)
```

OUTPUT:



```
import joblib

# Save the RandomForest model
joblib.dump(rf_model, "best_model.pkl")

# Load the saved model
loaded_model = joblib.load("best_model.pkl")

# Check if the loaded model performs the same
y_pred_loaded = loaded_model.predict(x_test)
print("Loaded Model Accuracy:", accuracy_score(y_test, y_pred_loaded))
```

Loaded Model Accuracy: 0.9988051063020187

Q6) Take the original data set and make another dataset by randomly picking 20 data points from the oil spill dataset and applying the saved model to the same.

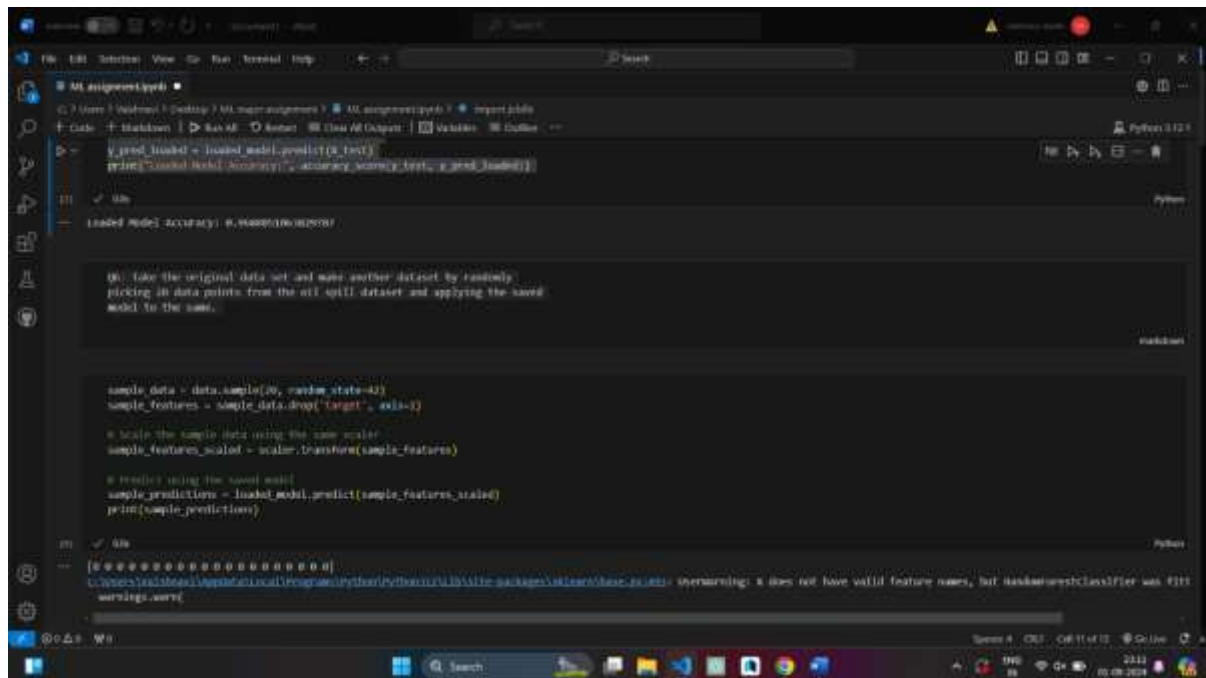
```
sample_data = data.sample(20, random_state=42)
sample_features = sample_data.drop("target", axis=1)

# Scale the sample data using the same scaler
sample_features_scaled = scaler.transform(sample_features)

# Predict using the saved model
```

Q6) Take the original data set and make another dataset by randomly picking 20 data points from the oil spill dataset and applying the saved model to the same.

INPUT:



```
Q6) Take the original data set and make another dataset by randomly
picking 20 data points from the oil spill dataset and applying the saved
model to the same.

sample_data = data.sample(20, random_state=42)
sample_features = sample_data.drop('target', axis=1)

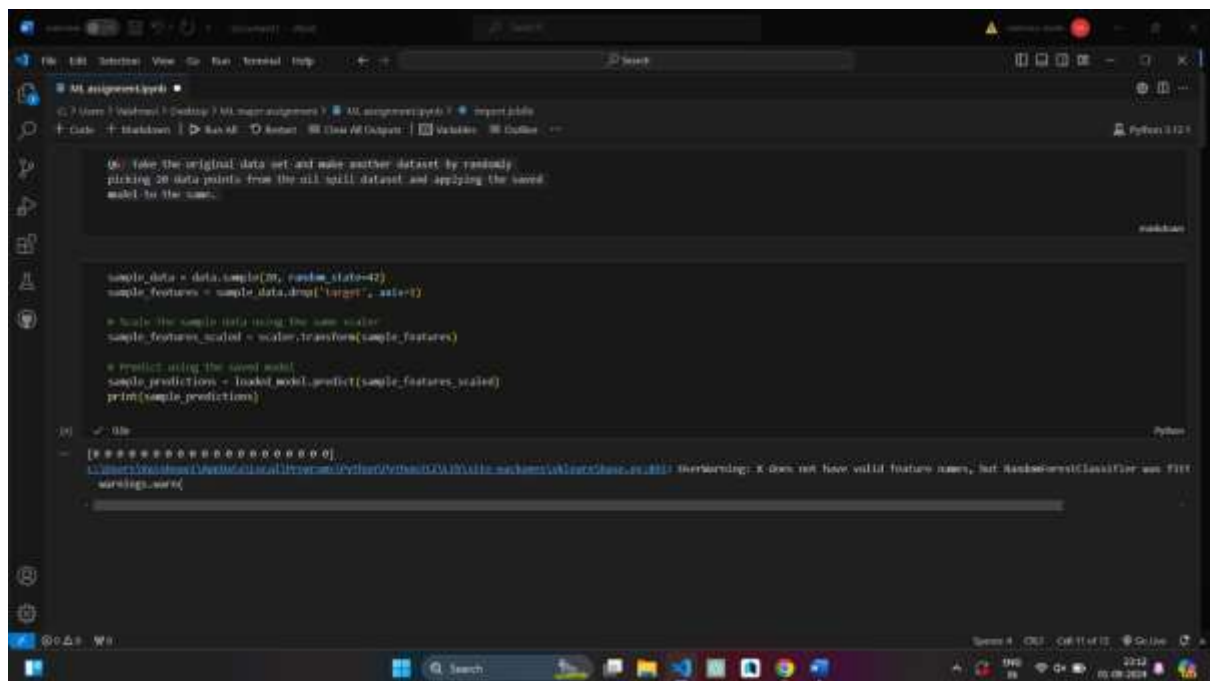
# Scale the sample data using the same scaler
sample_features_scaled = scaler.transform(sample_features)

# Predict using the saved model
sample_predictions = loaded_model.predict(sample_features_scaled)
print(sample_predictions)

loaded Model accuracy: 0.9448910604285714

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
C:\Users\ankush\AppData\Local\Programs\Python\Python311\Scripts\python.exe: warning: x does not have valid feature names, but RandomForestClassifier was fitted
warnings.warn()
```


OUTPUT:



The screenshot shows a Jupyter Notebook interface within a dark-themed VS Code editor. The notebook has a title bar 'ML assignment.ipynb' and a toolbar with icons for file operations, search, and execution. The code cell contains a Python script that performs the following steps:

- Imports `data` and `model` from a module named `ml_assignment`.
- Comments: "Take the original data set and make another dataset by randomly picking 20 data points from the all split dataset and applying the saved model to the same."
- Code: `sample_data = data.sample(20, random_state=42)`
- Code: `sample_features = sample_data.drop('target', axis=1)`
- Comments: "Scale the sample data using the same scaler"
- Code: `sample_features_scaled = scaler.transform(sample_features)`
- Comments: "Predict using the saved model"
- Code: `sample_predictions = loaded_model.predict(sample_features_scaled)`
- Code: `print(sample_predictions)`

The output cell shows a single line of text: `[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]`. Below this, a warning message is displayed: `Warning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names`. The bottom status bar indicates the file is 'Untitled', the editor is in 'Normal' mode, and the Python version is 'Python 3.12.4'.