```
from google.colab import files
upload = files.upload()
```

Choose Files mushroom.csv

- **mushroom.csv**(text/csv) - 3171031 bytes, last modified: 4/18/2024 - 100% done

Saving mushroom.csv to mushroom.csv

```
from google.colab import files
upload = files.upload()
```

Choose Files dataPrep.py

- **dataPrep.py**(text/x-python) - 468 bytes, last modified: 2/6/2025 - 100% done

Saving dataPrep.py to dataPrep.py

```
import pandas as pd
df = pd.read_csv('mushroom.csv')
df
```

	cap-diameter	cap-shape	gill-attachment	gill-color	stem-height	stem-width	stem-color	season	class
0	1372	2	2	10	3.807467	1545	11	1.804273	1
1	1461	2	2	10	3.807467	1557	11	1.804273	1
2	1371	2	2	10	3.612496	1566	11	1.804273	1
3	1261	6	2	10	3.787572	1566	11	1.804273	1
4	1305	6	2	10	3.711971	1464	11	0.943195	1
...	...	...	...	...	...	...	...	...	...
54030	73	5	3	2	0.887740	569	12	0.943195	1
54031	82	2	3	2	1.186164	490	12	0.943195	1
54032	82	5	3	2	0.915593	584	12	0.888450	1
54033	79	2	3	2	1.034963	491	12	0.888450	1
54034	72	5	3	2	1.158311	492	12	0.888450	1

54035 rows × 9 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
from dataPrep import dataPreperation
df2 = dataPreperation(df)
df2
```

	cap-diameter	cap-shape	gill-attachment	gill-color	stem-height	stem-width	stem-color	season	class
11	642	6	2	10	0.286062	1311	11	0.943195	1
12	814	4	2	10	1.189292	1681	11	0.943195	1
13	550	4	2	10	0.548675	1220	11	0.888450	1
14	606	6	2	10	0.254230	1239	11	0.943195	1
15	721	6	2	10	0.950553	1445	11	0.943195	1
...	...	...	...	...	...	...	...	...	...
54030	73	5	3	2	0.887740	569	12	0.943195	1
54031	82	2	3	2	1.186164	490	12	0.943195	1
54032	82	5	3	2	0.915593	584	12	0.888450	1
54033	79	2	3	2	1.034963	491	12	0.888450	1
54034	72	5	3	2	1.158311	492	12	0.888450	1

44504 rows × 9 columns

Next steps: Generate code with df2 | View recommended plots | New interactive sheet

```
df3 = dataPreperation(df2)
df3
```

	cap-diameter	cap-shape	gill-attachment	gill-color	stem-height	stem-width	stem-color	season	class
11	642	6	2	10	0.286062	1311	11	0.943195	1
12	814	4	2	10	1.189292	1681	11	0.943195	1
13	550	4	2	10	0.548675	1220	11	0.888450	1
14	606	6	2	10	0.254230	1239	11	0.943195	1
15	721	6	2	10	0.950553	1445	11	0.943195	1
...	...	...	...	...	...	...	...	...	...
54030	73	5	3	2	0.887740	569	12	0.943195	1
54031	82	2	3	2	1.186164	490	12	0.943195	1
54032	82	5	3	2	0.915593	584	12	0.888450	1
54033	79	2	3	2	1.034963	491	12	0.888450	1
54034	72	5	3	2	1.158311	492	12	0.888450	1

43689 rows × 9 columns

Next steps: Generate code with df3 View recommended plots New interactive sheet

```
df4 = dataPreperation(df3)
df4
```

	cap-diameter	cap-shape	gill-attachment	gill-color	stem-height	stem-width	stem-color	season	class
11	642	6	2	10	0.286062	1311	11	0.943195	1
12	814	4	2	10	1.189292	1681	11	0.943195	1
13	550	4	2	10	0.548675	1220	11	0.888450	1
14	606	6	2	10	0.254230	1239	11	0.943195	1
15	721	6	2	10	0.950553	1445	11	0.943195	1
...	...	...	...	...	...	...	...	...	...
54030	73	5	3	2	0.887740	569	12	0.943195	1
54031	82	2	3	2	1.186164	490	12	0.943195	1
54032	82	5	3	2	0.915593	584	12	0.888450	1
54033	79	2	3	2	1.034963	491	12	0.888450	1
54034	72	5	3	2	1.158311	492	12	0.888450	1

43444 rows × 9 columns

Next steps: Generate code with df4 View recommended plots New interactive sheet

```
df5 = dataPreperation(df4)
df5
```

	cap-diameter	cap-shape	gill-attachment	gill-color	stem-height	stem-width	stem-color	season	class
11	642	6	2	10	0.286062	1311	11	0.943195	1
12	814	4	2	10	1.189292	1681	11	0.943195	1
13	550	4	2	10	0.548675	1220	11	0.888450	1
14	606	6	2	10	0.254230	1239	11	0.943195	1
15	721	6	2	10	0.950553	1445	11	0.943195	1
...	...	...	...	...	...	...	...	...	...
54030	73	5	3	2	0.887740	569	12	0.943195	1
54031	82	2	3	2	1.186164	490	12	0.943195	1
54032	82	5	3	2	0.915593	584	12	0.888450	1
54033	79	2	3	2	1.034963	491	12	0.888450	1
54034	72	5	3	2	1.158311	492	12	0.888450	1

43348 rows × 9 columns

Next steps: Generate code with df5 View recommended plots New interactive sheet

```
df6 = dataPreperation(df5)
df6
```

	cap-diameter	cap-shape	gill-attachment	gill-color	stem-height	stem-width	stem-color	season	class
11	642	6	2	10	0.286062	1311	11	0.943195	1
12	814	4	2	10	1.189292	1681	11	0.943195	1
13	550	4	2	10	0.548675	1220	11	0.888450	1
14	606	6	2	10	0.254230	1239	11	0.943195	1
15	721	6	2	10	0.950553	1445	11	0.943195	1
...	...	...	...	...	...	...	...	...	...
54030	73	5	3	2	0.887740	569	12	0.943195	1
54031	82	2	3	2	1.186164	490	12	0.943195	1
54032	82	5	3	2	0.915593	584	12	0.888450	1
54033	79	2	3	2	1.034963	491	12	0.888450	1
54034	72	5	3	2	1.158311	492	12	0.888450	1

43338 rows × 9 columns

Next steps: Generate code with df6 View recommended plots New interactive sheet

```
df7 = dataPreperation(df6)
df7
```

	cap-diameter	cap-shape	gill-attachment	gill-color	stem-height	stem-width	stem-color	season	class
11	642	6	2	10	0.286062	1311	11	0.943195	1
12	814	4	2	10	1.189292	1681	11	0.943195	1
13	550	4	2	10	0.548675	1220	11	0.888450	1
14	606	6	2	10	0.254230	1239	11	0.943195	1
15	721	6	2	10	0.950553	1445	11	0.943195	1
...	...	...	...	...	...	...	...	...	...
54030	73	5	3	2	0.887740	569	12	0.943195	1
54031	82	2	3	2	1.186164	490	12	0.943195	1
54032	82	5	3	2	0.915593	584	12	0.888450	1
54033	79	2	3	2	1.034963	491	12	0.888450	1
54034	72	5	3	2	1.158311	492	12	0.888450	1

42787 rows × 9 columns

Next steps: Generate code with df7 View recommended plots New interactive sheet

```
df8 = dataPreperation(df7)
df8
```

	cap-diameter	cap-shape	gill-attachment	gill-color	stem-height	stem-width	stem-color	season	class
11	642	6	2	10	0.286062	1311	11	0.943195	1
12	814	4	2	10	1.189292	1681	11	0.943195	1
13	550	4	2	10	0.548675	1220	11	0.888450	1
14	606	6	2	10	0.254230	1239	11	0.943195	1
15	721	6	2	10	0.950553	1445	11	0.943195	1
...	...	...	...	...	...	...	...	...	...
54030	73	5	3	2	0.887740	569	12	0.943195	1
54031	82	2	3	2	1.186164	490	12	0.943195	1
54032	82	5	3	2	0.915593	584	12	0.888450	1
54033	79	2	3	2	1.034963	491	12	0.888450	1
54034	72	5	3	2	1.158311	492	12	0.888450	1

42633 rows × 9 columns

Next steps: Generate code with df8 View recommended plots New interactive sheet

```
df9 = dataPreperation(df8)
df9
```

	cap-diameter	cap-shape	gill-attachment	gill-color	stem-height	stem-width	stem-color	season	class
11	642	6	2	10	0.286062	1311	11	0.943195	1
12	814	4	2	10	1.189292	1681	11	0.943195	1
13	550	4	2	10	0.548675	1220	11	0.888450	1
14	606	6	2	10	0.254230	1239	11	0.943195	1
15	721	6	2	10	0.950553	1445	11	0.943195	1
...	...	...	...	...	...	...	...	...	...
54030	73	5	3	2	0.887740	569	12	0.943195	1
54031	82	2	3	2	1.186164	490	12	0.943195	1
54032	82	5	3	2	0.915593	584	12	0.888450	1
54033	79	2	3	2	1.034963	491	12	0.888450	1
54034	72	5	3	2	1.158311	492	12	0.888450	1

42598 rows × 9 columns

Next steps: Generate code with df9 View recommended plots New interactive sheet

```
df10 = dataPreperation(df9)
df10
```

	cap-diameter	cap-shape	gill-attachment	gill-color	stem-height	stem-width	stem-color	season	class
11	642	6	2	10	0.286062	1311	11	0.943195	1
12	814	4	2	10	1.189292	1681	11	0.943195	1
13	550	4	2	10	0.548675	1220	11	0.888450	1
14	606	6	2	10	0.254230	1239	11	0.943195	1
15	721	6	2	10	0.950553	1445	11	0.943195	1
...	...	...	...	...	...	...	...	...	...
54030	73	5	3	2	0.887740	569	12	0.943195	1
54031	82	2	3	2	1.186164	490	12	0.943195	1
54032	82	5	3	2	0.915593	584	12	0.888450	1
54033	79	2	3	2	1.034963	491	12	0.888450	1
54034	72	5	3	2	1.158311	492	12	0.888450	1

42595 rows × 9 columns

Next steps: Generate code with df10 View recommended plots New interactive sheet

```
df11 = dataPreperation(df10)
df11
```

	cap-diameter	cap-shape	gill-attachment	gill-color	stem-height	stem-width	stem-color	season	class
11	642	6	2	10	0.286062	1311	11	0.943195	1
12	814	4	2	10	1.189292	1681	11	0.943195	1
13	550	4	2	10	0.548675	1220	11	0.888450	1
14	606	6	2	10	0.254230	1239	11	0.943195	1
15	721	6	2	10	0.950553	1445	11	0.943195	1
...	...	...	...	...	...	...	...	...	...
54030	73	5	3	2	0.887740	569	12	0.943195	1
54031	82	2	3	2	1.186164	490	12	0.943195	1
54032	82	5	3	2	0.915593	584	12	0.888450	1
54033	79	2	3	2	1.034963	491	12	0.888450	1
54034	72	5	3	2	1.158311	492	12	0.888450	1

42595 rows × 9 columns

Next steps: Generate code with df11 View recommended plots New interactive sheet

```
X = df11.drop('class', axis=1)
y = df11['class']
```

```
y.value_counts()
```

class	count
1	23870
0	18725

dtype: int64

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, random_state=30)
```

```
def runModel(model, X_train, X_test, y_train, y_test):
  model.fit(X_train, y_train)

  y_pred = model.predict(X_test)
  y_train_pred = model.predict(X_train)

  acc_test = accuracy_score(y_test, y_pred)
  acc_train = accuracy_score(y_train, y_train_pred)

  y_pred_proba = model.predict_proba(X_test)

  cm = confusion_matrix(y_test, y_pred)
  # cr = classification_report(y_test, y_pred)
  print(f'Train Accuracy: {acc_train}')
  print(f'Test Accuracy: {acc_test}')
  print(cm)
  return acc_test, acc_train, cm, y_pred_proba, y_pred
```

```
from sklearn.linear_model import LogisticRegression as LR
from sklearn.tree import DecisionTreeClassifier as DT
from xgboost import XGBClassifier as XGB
from sklearn.neighbors import KNeighborsClassifier as KNN
from sklearn.ensemble import RandomForestClassifier as RF
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report


# model = LogisticRegression(max_iter= 10000)

models_arr = [LR(max_iter=10000), DT(), XGB(), KNN(), RF()]
for model in models_arr:
  print(model.__class__.__name__)
```

```
runModel(model, X_train, X_test, y_train, y_test)
acc_test, acc_train, cm, y_pred_proba, y_pred = runModel(model, X_train, X_test, y_train, y_test)
```

```
LogisticRegression
Train Accuracy: 0.6397236383149987
Test Accuracy: 0.6373738164175601
[[2921 2740]
 [1894 5224]]
Train Accuracy: 0.6397236383149987
Test Accuracy: 0.6373738164175601
[[2921 2740]
 [1894 5224]]
DecisionTreeClassifier
Train Accuracy: 1.0
Test Accuracy: 0.9701854605211675
[[5451  210]
 [ 171 6947]]
Train Accuracy: 1.0
Test Accuracy: 0.9689334063698255
[[5446  215]
 [ 182 6936]]
XGBClassifier
Train Accuracy: 0.9946337536892943
Test Accuracy: 0.9854448704906488
[[5567   94]
 [  92 7026]]
Train Accuracy: 0.9946337536892943
Test Accuracy: 0.9854448704906488
[[5567   94]
 [  92 7026]]
KNeighborsClassifier
Train Accuracy: 0.8135564797424202
Test Accuracy: 0.7082713827373034
[[3769 1892]
 [1836 5282]]
Train Accuracy: 0.8135564797424202
Test Accuracy: 0.7082713827373034
[[3769 1892]
 [1836 5282]]
RandomForestClassifier
Train Accuracy: 1.0
Test Accuracy: 0.986540417873073
[[5570   91]
 [  81 7037]]
Train Accuracy: 1.0
Test Accuracy: 0.9873229517176618
[[5582   79]
 [  83 7035]]
```

```
df11
```

	cap-diameter	cap-shape	gill-attachment	gill-color	stem-height	stem-width	stem-color	season	class
11	642	6	2	10	0.286062	1311	11	0.943195	1
12	814	4	2	10	1.189292	1681	11	0.943195	1
13	550	4	2	10	0.548675	1220	11	0.888450	1
14	606	6	2	10	0.254230	1239	11	0.943195	1
15	721	6	2	10	0.950553	1445	11	0.943195	1
...	...	...	...	...	...	...	...	...	...
54030	73	5	3	2	0.887740	569	12	0.943195	1
54031	82	2	3	2	1.186164	490	12	0.943195	1
54032	82	5	3	2	0.915593	584	12	0.888450	1
54033	79	2	3	2	1.034963	491	12	0.888450	1
54034	72	5	3	2	1.158311	492	12	0.888450	1

42595 rows × 9 columns

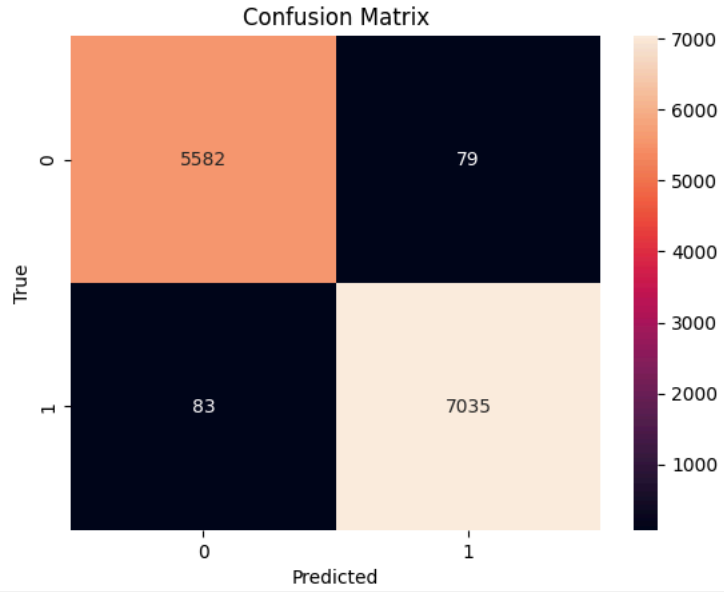
Next steps: Generate code with df11 View recommended plots New interactive sheet

```
# prompt: visualize cm

import seaborn as sns
import matplotlib.pyplot as plt

# Assuming 'cm' is your confusion matrix from the previous code
sns.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
```

```
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```




```
from sklearn.preprocessing import MinMaxScaler
minmaxscaller = MinMaxScaler()
df11_scaled = minmaxscaller.fit_transform(df11)
df11_scaled = pd.DataFrame(df11_scaled, columns=df11.columns)
df11_scaled
```

	cap-diameter	cap-shape	gill-attachment	gill-color	stem-height	stem-width	stem-color	season	class
0	0.448324	1.000000	0.333333	0.909091	0.145379	0.454545	0.916667	1.0	1.0
1	0.568436	0.666667	0.333333	0.909091	0.605092	0.582929	0.916667	1.0	1.0
2	0.384078	0.666667	0.333333	0.909091	0.279040	0.422970	0.916667	0.0	1.0
3	0.423184	1.000000	0.333333	0.909091	0.129177	0.429563	0.916667	1.0	1.0
4	0.503492	1.000000	0.333333	0.909091	0.483582	0.501041	0.916667	1.0	1.0
...	...	...	...	...	...	...	...	...	...
42590	0.050978	0.833333	0.500000	0.181818	0.451613	0.197085	1.000000	1.0	1.0
42591	0.057263	0.333333	0.500000	0.181818	0.603500	0.169674	1.000000	1.0	1.0
42592	0.057263	0.833333	0.500000	0.181818	0.465789	0.202290	1.000000	0.0	1.0
42593	0.055168	0.333333	0.500000	0.181818	0.526544	0.170021	1.000000	0.0	1.0
42594	0.050279	0.833333	0.500000	0.181818	0.589324	0.170368	1.000000	0.0	1.0

42595 rows × 9 columns

Next steps: Generate code with df11_scaled | View recommended plots | New interactive sheet

```
X_scaled = df11_scaled.drop('class', axis=1)
y_scaled = df11_scaled['class']
```

```
X_train_scaled, X_test_scaled, y_train_scaled, y_test_scaled = train_test_split(X_scaled, y_scaled, train_size=0.7, random_state=30)
```

```
X_train_scaled.shape, X_test_scaled.shape, y_train_scaled.shape, y_test_scaled.shape
```

```
((29816, 8), (12779, 8), (29816,), (12779,))
```

```
for model in models_arr:
  print(model.__class__.__name__)
  runModel(model, X_train_scaled, X_test_scaled, y_train_scaled, y_test_scaled)
```

```
LogisticRegression
Train Accuracy: 0.6393882479205796
Test Accuracy: 0.637452069802019
[[2925 2736]
 [1897 5221]]
```

```
DecisionTreeClassifier
Train Accuracy: 1.0
Test Accuracy: 0.9698724469833321
[[5450  211]
 [ 174 6944]]
XGBClassifier
Train Accuracy: 0.9946337536892943
Test Accuracy: 0.9854448704906488
[[5567   94]
 [  92 7026]]
KNeighborsClassifier
Train Accuracy: 0.9894687416152401
Test Accuracy: 0.9827842554190469
[[5538  123]
 [  97 7021]]
RandomForestClassifier
Train Accuracy: 1.0
Test Accuracy: 0.9871664449487441
[[5569   92]
 [  72 7046]]
```

```
y_train_scaled.value_counts()
```

class	count
1.0	16752
0.0	13064

dtype: int64

```
from imblearn.over_sampling import SMOTE
smote = SMOTE()
X_train_smote, y_train_smote = smote.fit_resample(X_train_scaled, y_train_scaled)
```

```
y_train_smote.value_counts()
```

class	count
0.0	16752
1.0	16752

dtype: int64

```
for model in models_arr:
  print(model.__class__.__name__)
  runModel(model, X_train_smote, X_test_scaled, y_train_smote, y_test_scaled)
```

```
LogisticRegression
Train Accuracy: 0.6423412129894938
Test Accuracy: 0.6384693637999843
[[3672 1989]
 [2631 4487]]
DecisionTreeClassifier
Train Accuracy: 1.0
Test Accuracy: 0.9744111432819469
[[5479  182]
 [ 145 6973]]
XGBClassifier
Train Accuracy: 0.9951946036294174
Test Accuracy: 0.9860708975663197
[[5582   79]
 [  99 7019]]
KNeighborsClassifier
Train Accuracy: 0.9905981375358166
Test Accuracy: 0.9828625088035058
[[5555  106]
 [ 113 7005]]
RandomForestClassifier
Train Accuracy: 1.0
Test Accuracy: 0.9870881915642852
[[5585   76]
 [  89 7029]]
```

Generate | visualise the randomforestclassifier cm and write the train and test accuracy | Close

1 of 1 | Use code with caution

```
# prompt: visualise the randomforestclassifier cm and write the train and test accuracy
```

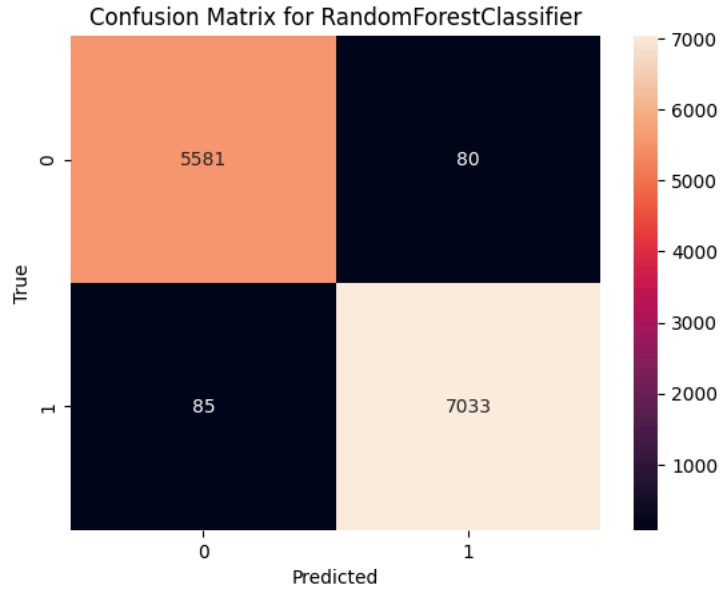
```
# prompt: visualise the RandomForestClassifier cm and write the train and test accuracy

# Assuming 'cm' is your confusion matrix from the previous code
# You need to call runModel for RandomForestClassifier and capture its output
# Example:
model = RF()
acc_test, acc_train, cm, y_pred_proba, y_pred = runModel(model, X_train, X_test, y_train, y_test)

sns.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix for RandomForestClassifier')
plt.show()

print(f'Train Accuracy for RandomForestClassifier: {acc_train}')
print(f'Test Accuracy for RandomForestClassifier: {acc_test}')
```

```
Train Accuracy: 1.0
Test Accuracy: 0.9870881915642852
[[5581   80]
 [  85 7033]]
```




```
Train Accuracy for RandomForestClassifier: 1.0
Test Accuracy for RandomForestClassifier: 0.9870881915642852
```

```
df_smote = pd.concat([X_train_smote, y_train_smote], axis=1)
df_smote
```

	cap-diameter	cap-shape	gill-attachment	gill-color	stem-height	stem-width	stem-color	season	class
0	0.729749	1.000000	0.000000	0.909091	0.082599	0.860861	0.500000	1.0	0.0
1	0.209497	0.333333	0.166667	0.454545	0.552871	0.192575	0.500000	0.0	1.0
2	0.646648	1.000000	0.333333	0.636364	0.566614	0.588480	0.916667	1.0	0.0
3	0.151536	0.166667	0.000000	0.454545	0.337770	0.095767	1.000000	0.0	1.0
4	0.837291	0.333333	1.000000	0.454545	0.161580	0.737335	0.500000	0.0	0.0
...	...	...	...	...	...	...	...	...	...
33499	0.502108	0.833333	0.833333	0.636364	0.191351	0.589266	0.833333	1.0	0.0
33500	0.699389	1.000000	0.333333	0.636364	0.588786	0.630264	0.916667	1.0	0.0
33501	0.517636	0.666667	0.000000	0.909091	0.353267	0.453976	0.250000	1.0	0.0
33502	0.407532	1.000000	0.166667	0.000000	0.032293	0.429930	0.916667	1.0	0.0
33503	0.249683	1.000000	0.166667	0.545455	0.013697	0.247544	0.583333	1.0	0.0

33504 rows × 9 columns

Next steps: Generate code with df_smote | View recommended plots | New interactive sheet

```
df12 = dataPreperation(df_smote)
df12
```

	cap-diameter	cap-shape	gill-attachment	gill-color	stem-height	stem-width	stem-color	season	class
0	0.729749	1.000000	0.000000	0.909091	0.082599	0.860861	0.500000	1.0	0.0
1	0.209497	0.333333	0.166667	0.454545	0.552871	0.192575	0.500000	0.0	1.0
2	0.646648	1.000000	0.333333	0.636364	0.566614	0.588480	0.916667	1.0	0.0
3	0.151536	0.166667	0.000000	0.454545	0.337770	0.095767	1.000000	0.0	1.0
4	0.837291	0.333333	1.000000	0.454545	0.161580	0.737335	0.500000	0.0	0.0
...	...	...	...	...	...	...	...	...	...
33499	0.502108	0.833333	0.833333	0.636364	0.191351	0.589266	0.833333	1.0	0.0
33500	0.699389	1.000000	0.333333	0.636364	0.588786	0.630264	0.916667	1.0	0.0
33501	0.517636	0.666667	0.000000	0.909091	0.353267	0.453976	0.250000	1.0	0.0
33502	0.407532	1.000000	0.166667	0.000000	0.032293	0.429930	0.916667	1.0	0.0
33503	0.249683	1.000000	0.166667	0.545455	0.013697	0.247544	0.583333	1.0	0.0

33343 rows × 9 columns

Next steps: Generate code with df12 View recommended plots New interactive sheet

```
df13 = dataPreperation(df12)
df13
```

	cap-diameter	cap-shape	gill-attachment	gill-color	stem-height	stem-width	stem-color	season	class
0	0.729749	1.000000	0.000000	0.909091	0.082599	0.860861	0.500000	1.0	0.0
1	0.209497	0.333333	0.166667	0.454545	0.552871	0.192575	0.500000	0.0	1.0
2	0.646648	1.000000	0.333333	0.636364	0.566614	0.588480	0.916667	1.0	0.0
3	0.151536	0.166667	0.000000	0.454545	0.337770	0.095767	1.000000	0.0	1.0
4	0.837291	0.333333	1.000000	0.454545	0.161580	0.737335	0.500000	0.0	0.0
...	...	...	...	...	...	...	...	...	...
33499	0.502108	0.833333	0.833333	0.636364	0.191351	0.589266	0.833333	1.0	0.0
33500	0.699389	1.000000	0.333333	0.636364	0.588786	0.630264	0.916667	1.0	0.0
33501	0.517636	0.666667	0.000000	0.909091	0.353267	0.453976	0.250000	1.0	0.0
33502	0.407532	1.000000	0.166667	0.000000	0.032293	0.429930	0.916667	1.0	0.0
33503	0.249683	1.000000	0.166667	0.545455	0.013697	0.247544	0.583333	1.0	0.0

33287 rows × 9 columns

Next steps: Generate code with df13 View recommended plots New interactive sheet

```
df14 = dataPreperation(df13)
df14
```

	cap-diameter	cap-shape	gill-attachment	gill-color	stem-height	stem-width	stem-color	season	class
0	0.729749	1.000000	0.000000	0.909091	0.082599	0.860861	0.500000	1.0	0.0
1	0.209497	0.333333	0.166667	0.454545	0.552871	0.192575	0.500000	0.0	1.0
2	0.646648	1.000000	0.333333	0.636364	0.566614	0.588480	0.916667	1.0	0.0
3	0.151536	0.166667	0.000000	0.454545	0.337770	0.095767	1.000000	0.0	1.0
4	0.837291	0.333333	1.000000	0.454545	0.161580	0.737335	0.500000	0.0	0.0
...	...	...	...	...	...	...	...	...	...
33499	0.502108	0.833333	0.833333	0.636364	0.191351	0.589266	0.833333	1.0	0.0
33500	0.699389	1.000000	0.333333	0.636364	0.588786	0.630264	0.916667	1.0	0.0
33501	0.517636	0.666667	0.000000	0.909091	0.353267	0.453976	0.250000	1.0	0.0
33502	0.407532	1.000000	0.166667	0.000000	0.032293	0.429930	0.916667	1.0	0.0
33503	0.249683	1.000000	0.166667	0.545455	0.013697	0.247544	0.583333	1.0	0.0

33251 rows × 9 columns

Next steps: Generate code with df14 View recommended plots New interactive sheet

```
df15 = dataPreperation(df14)
df15
```

	cap-diameter	cap-shape	gill-attachment	gill-color	stem-height	stem-width	stem-color	season	class
0	0.729749	1.000000	0.000000	0.909091	0.082599	0.860861	0.500000	1.0	0.0
1	0.209497	0.333333	0.166667	0.454545	0.552871	0.192575	0.500000	0.0	1.0
2	0.646648	1.000000	0.333333	0.636364	0.566614	0.588480	0.916667	1.0	0.0
3	0.151536	0.166667	0.000000	0.454545	0.337770	0.095767	1.000000	0.0	1.0
4	0.837291	0.333333	1.000000	0.454545	0.161580	0.737335	0.500000	0.0	0.0
...	...	...	...	...	...	...	...	...	...
33499	0.502108	0.833333	0.833333	0.636364	0.191351	0.589266	0.833333	1.0	0.0
33500	0.699389	1.000000	0.333333	0.636364	0.588786	0.630264	0.916667	1.0	0.0
33501	0.517636	0.666667	0.000000	0.909091	0.353267	0.453976	0.250000	1.0	0.0
33502	0.407532	1.000000	0.166667	0.000000	0.032293	0.429930	0.916667	1.0	0.0
33503	0.249683	1.000000	0.166667	0.545455	0.013697	0.247544	0.583333	1.0	0.0

33250 rows × 9 columns

Next steps: Generate code with df15 View recommended plots New interactive sheet

```
df16 = dataPreperation(df15)
df16
```

	cap-diameter	cap-shape	gill-attachment	gill-color	stem-height	stem-width	stem-color	season	class
0	0.729749	1.000000	0.000000	0.909091	0.082599	0.860861	0.500000	1.0	0.0
1	0.209497	0.333333	0.166667	0.454545	0.552871	0.192575	0.500000	0.0	1.0
2	0.646648	1.000000	0.333333	0.636364	0.566614	0.588480	0.916667	1.0	0.0
3	0.151536	0.166667	0.000000	0.454545	0.337770	0.095767	1.000000	0.0	1.0
4	0.837291	0.333333	1.000000	0.454545	0.161580	0.737335	0.500000	0.0	0.0
...	...	...	...	...	...	...	...	...	...
33499	0.502108	0.833333	0.833333	0.636364	0.191351	0.589266	0.833333	1.0	0.0
33500	0.699389	1.000000	0.333333	0.636364	0.588786	0.630264	0.916667	1.0	0.0
33501	0.517636	0.666667	0.000000	0.909091	0.353267	0.453976	0.250000	1.0	0.0
33502	0.407532	1.000000	0.166667	0.000000	0.032293	0.429930	0.916667	1.0	0.0
33503	0.249683	1.000000	0.166667	0.545455	0.013697	0.247544	0.583333	1.0	0.0

33250 rows × 9 columns

Next steps: Generate code with df16 View recommended plots New interactive sheet

```
X = df16.drop('class', axis=1)
y = df16['class']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, random_state=30)
```

```
models_output = []
for model in models_arr:
  print(model.__class__.__name__)
  acc_test, acc_train, cm, y_pred_proba, y_pred=  runModel(model, X_train, X_test, y_train, y_test)
  models_output.append([model.__class__.__name__, acc_test, acc_train, cm, y_pred_proba,y_pred])
```

```
LogisticRegression
Train Accuracy: 0.6435660580021483
Test Accuracy: 0.6368922305764411
[[3348 1627]
 [1995 3005]]
DecisionTreeClassifier
Train Accuracy: 1.0
Test Accuracy: 0.9711278195488722
[[4825  150]
 [ 138 4862]]
XGBClassifier
Train Accuracy: 0.9957894736842106
Test Accuracy: 0.9884711779448622
```

```
[[4927   48]
 [  67 4933]]
KNeighborsClassifier
Train Accuracy: 0.9893018259935553
Test Accuracy: 0.983358395989975
[[4891   84]
 [  82 4918]]
RandomForestClassifier
Train Accuracy: 1.0
Test Accuracy: 0.9895739348370928
[[4929   46]
 [  58 4942]]
```

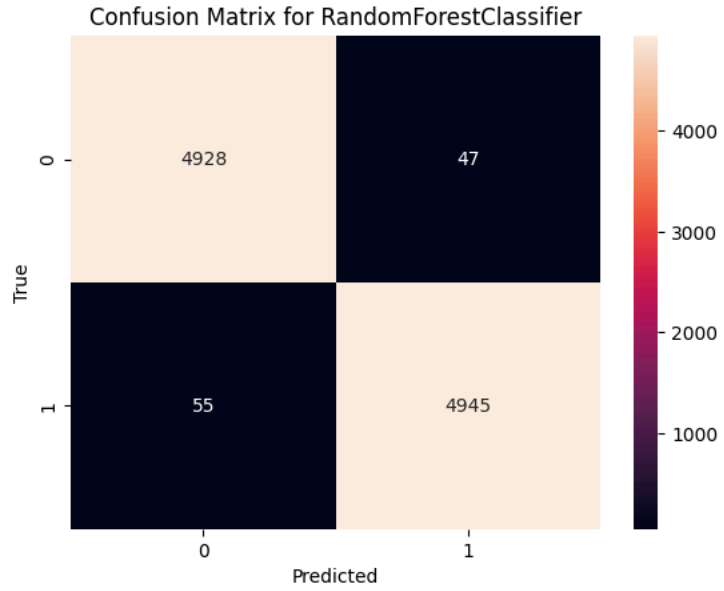
```
# prompt: visualise the randomforestclassifier cm and write the train and test accuracy

# Assuming 'cm' is your confusion matrix from the previous code
# You need to call runModel for RandomForestClassifier and capture its output
# Example:
model = RF()
acc_test, acc_train, cm, y_pred_proba, y_pred = runModel(model, X_train, X_test, y_train, y_test)

sns.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix for RandomForestClassifier')
plt.show()

print(f'Train Accuracy for RandomForestClassifier: {acc_train}')
print(f'Test Accuracy for RandomForestClassifier: {acc_test}')
```

```
Train Accuracy: 1.0
Test Accuracy: 0.9897744360902255
[[4928   47]
 [  55 4945]]
```




```
Train Accuracy for RandomForestClassifier: 1.0
Test Accuracy for RandomForestClassifier: 0.9897744360902255
```

```
models_output[0][4]
```

```
array([[0.44418789, 0.55581211],
       [0.33487222, 0.66512778],
       [0.39654844, 0.60345156],
       ...,
       [0.75881156, 0.24118844],
       [0.52548647, 0.47451353],
       [0.25316302, 0.74683698]])
```

```
models_output[0][5]
```

```
array([1., 1., 1., ..., 0., 0., 1.])
```

```
df_y_pred = pd.DataFrame(models_output[2][5], columns=['Prediction'])
df_y_pred
```

	Prediction
0	0
1	1
2	1
3	1
4	1
...	...
9970	0
9971	0
9972	1
9973	0
9974	1

9975 rows × 1 columns

Next Generate code with df_y_pred View recommended plots New interactive sheet

```
df_pred_proba = pd.DataFrame(models_output[2][4], columns=['0', '1'])
df_pred_proba
```

	0	1
0	0.945217	0.054783
1	0.076579	0.923421
2	0.000010	0.999990
3	0.008000	0.992000
4	0.000188	0.999812
...	...	...
9970	0.999559	0.000441
9971	0.997815	0.002185
9972	0.271758	0.728242
9973	0.999607	0.000393
9974	0.000339	0.999661

9975 rows × 2 columns

Next steps: Generate code with df_pred_proba View recommended plots New interactive sheet

```
df_both = pd.concat([df_y_pred, df_pred_proba], axis=1)
df_both
```

	Prediction	0	1
0	0	0.945217	0.054783
1	1	0.076579	0.923421
2	1	0.000010	0.999990
3	1	0.008000	0.992000
4	1	0.000188	0.999812
...	...	...	...
9970	0	0.999559	0.000441
9971	0	0.997815	0.002185
9972	1	0.271758	0.728242
9973	0	0.999607	0.000393
9974	1	0.000339	0.999661

9975 rows × 3 columns