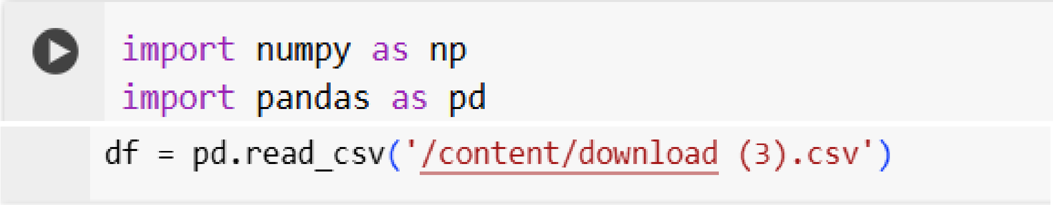
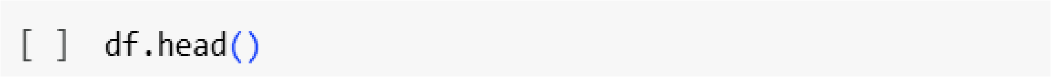
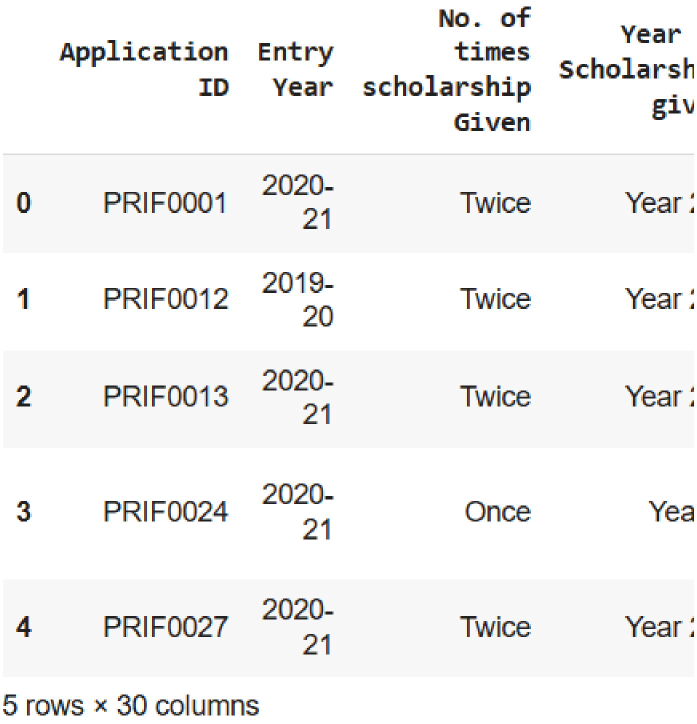
SCHOLARSHIP ELIGIBILITY CLASSIFICATION MODEL – ML



```
import numpy as np
import pandas as pd
df = pd.read_csv('/content/download (3).csv')
```

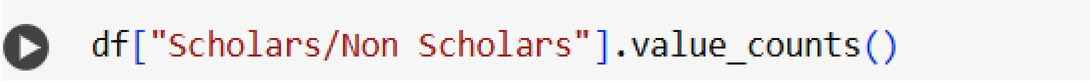


```
[ ] df.head()
```

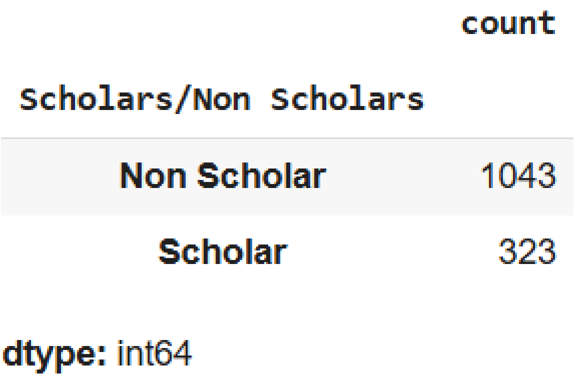


	Application ID	Entry Year	No. of times scholarship Given	Year Scholarsh giv
0	PRIF0001	2020-21	Twice	Year :
1	PRIF0012	2019-20	Twice	Year :
2	PRIF0013	2020-21	Twice	Year :
3	PRIF0024	2020-21	Once	Yea
4	PRIF0027	2020-21	Twice	Year :

5 rows × 30 columns



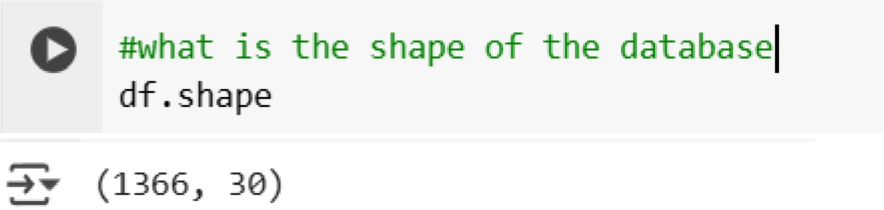
```
df["Scholars/Non Scholars"].value_counts()
```



Scholars/Non Scholars	count
Non Scholar	1043
Scholar	323

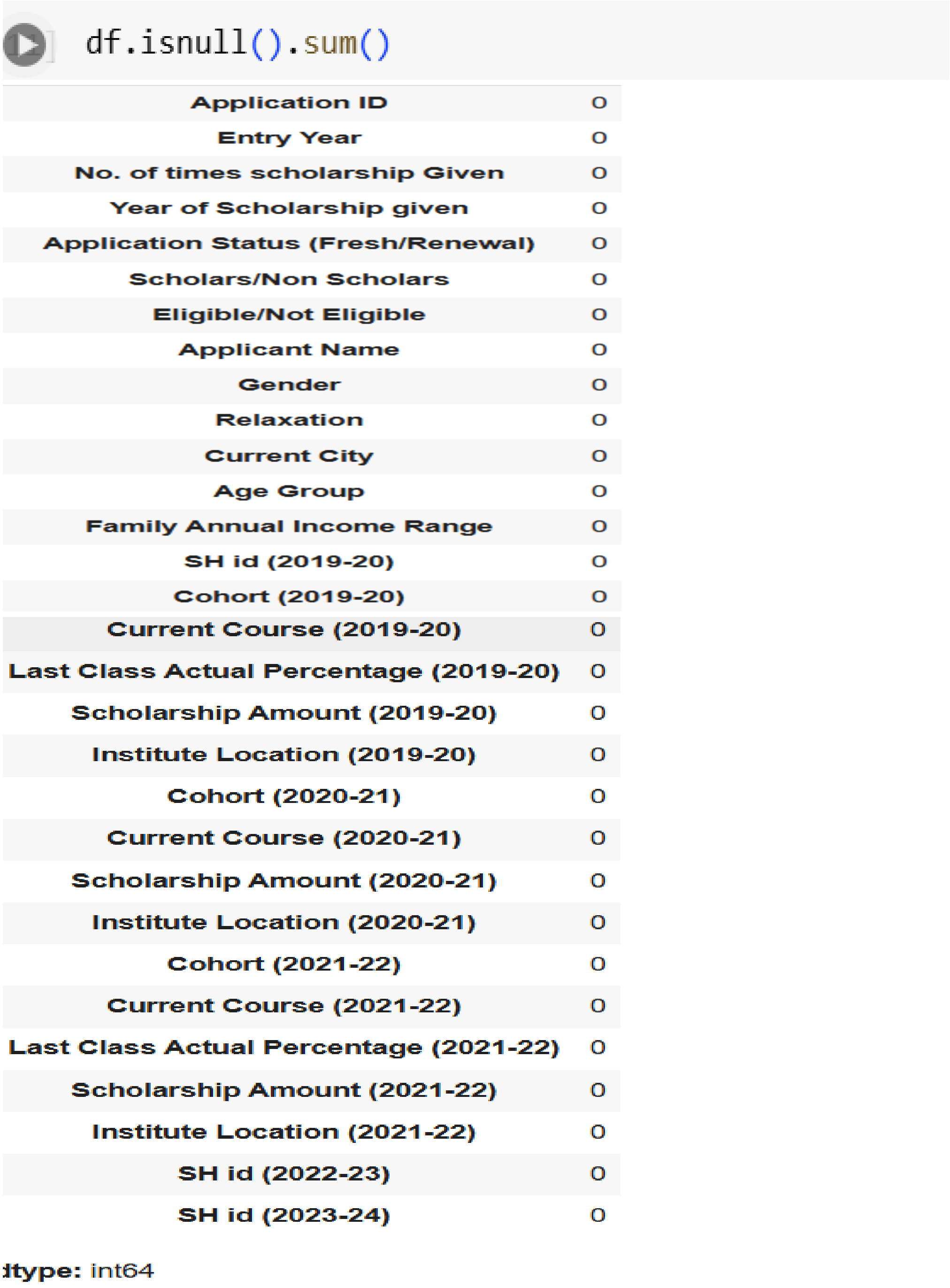
dtype: int64

ASK 6 QUESTIONS



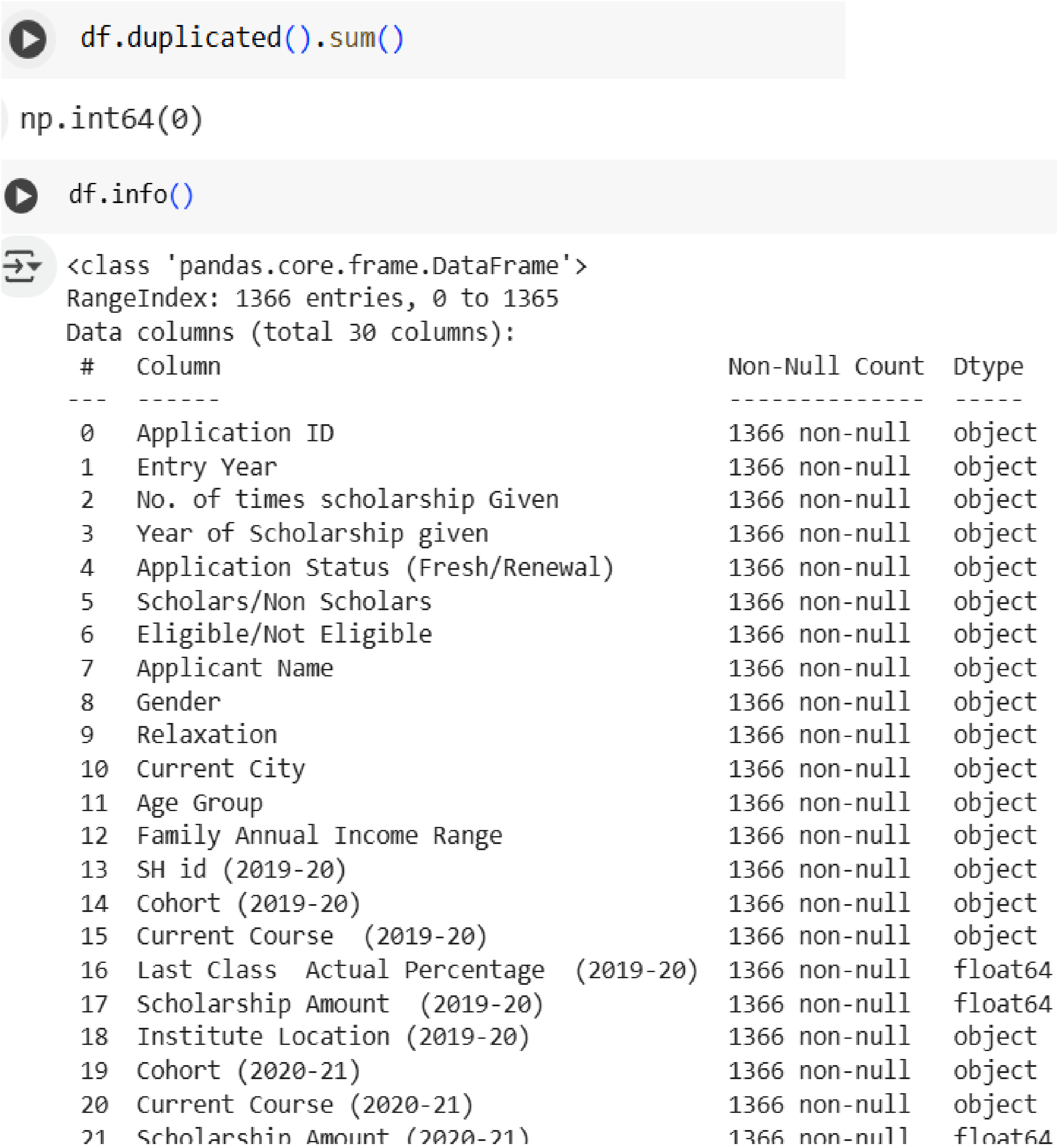
```
#what is the shape of the database
df.shape
```

```
(1366, 30)
```



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

```
                     Application ID  0
                         Entry Year  0
     No. of times scholarship Given  0
          Year of Scholarship given  0
Application Status (Fresh/Renewal)   0
            Scholars/Non Scholars    0
             Eligible/Not Eligible   0
                    Applicant Name   0
                            Gender   0
                        Relaxation   0
                      Current City   0
                         Age Group   0
       Family Annual Income Range    0
                    SH id (2019-20)  0
                  Cohort (2019-20)   0
          Current Course (2019-20)   0
Last Class Actual Percentage (2019-20)  0
      Scholarship Amount (2019-20)   0
      Institute Location (2019-20)   0
                  Cohort (2020-21)   0
          Current Course (2020-21)   0
      Scholarship Amount (2020-21)   0
      Institute Location (2020-21)   0
                  Cohort (2021-22)   0
          Current Course (2021-22)   0
Last Class Actual Percentage (2021-22)  0
      Scholarship Amount (2021-22)   0
      Institute Location (2021-22)   0
                   SH id (2022-23)   0
                   SH id (2023-24)   0

dtype: int64
```

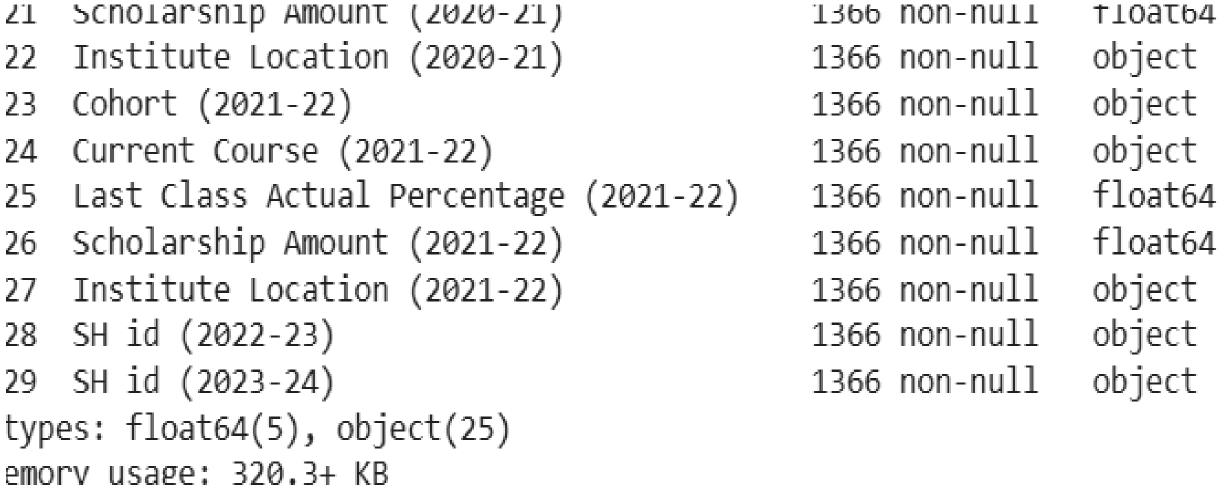


```
df.duplicated().sum()
```

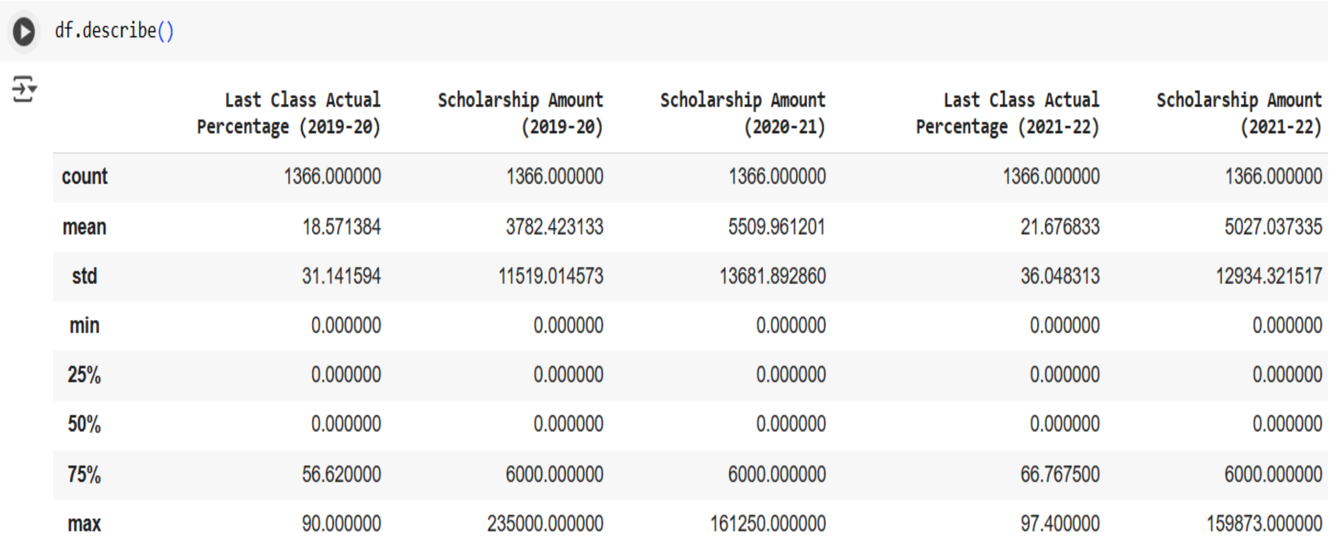
```
np.int64(0)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1366 entries, 0 to 1365
Data columns (total 30 columns):
 #   Column                                      Non-Null Count  Dtype  
---  ------                                      --------------  -----  
 0   Application ID                              1366 non-null   object 
 1   Entry Year                                  1366 non-null   object 
 2   No. of times scholarship Given              1366 non-null   object 
 3   Year of Scholarship given                   1366 non-null   object 
 4   Application Status (Fresh/Renewal)          1366 non-null   object 
 5   Scholars/Non Scholars                       1366 non-null   object 
 6   Eligible/Not Eligible                       1366 non-null   object 
 7   Applicant Name                              1366 non-null   object 
 8   Gender                                      1366 non-null   object 
 9   Relaxation                                  1366 non-null   object 
 10  Current City                                1366 non-null   object 
 11  Age Group                                   1366 non-null   object 
 12  Family Annual Income Range                  1366 non-null   object 
 13  SH id (2019-20)                             1366 non-null   object 
 14  Cohort (2019-20)                            1366 non-null   object 
 15  Current Course  (2019-20)                   1366 non-null   object 
 16  Last Class  Actual Percentage  (2019-20)    1366 non-null   float64
 17  Scholarship Amount  (2019-20)               1366 non-null   float64
 18  Institute Location (2019-20)                1366 non-null   object 
 19  Cohort (2020-21)                            1366 non-null   object 
 20  Current Course (2020-21)                    1366 non-null   object 
```



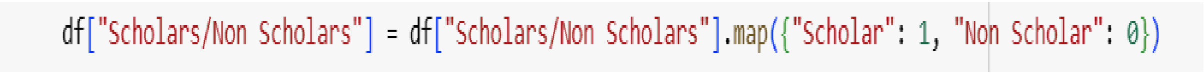
```
21  Scholarship Amount (2020-21)              1366 non-null   float64
22  Institute Location (2020-21)              1366 non-null   object 
23  Cohort (2021-22)                          1366 non-null   object 
24  Current Course (2021-22)                  1366 non-null   object 
25  Last Class Actual Percentage (2021-22)    1366 non-null   float64
26  Scholarship Amount (2021-22)              1366 non-null   float64
27  Institute Location (2021-22)              1366 non-null   object 
28  SH id (2022-23)                           1366 non-null   object 
29  SH id (2023-24)                           1366 non-null   object 
types: float64(5), object(25)
emory usage: 320.3+ KB
```



```
df.describe()
```

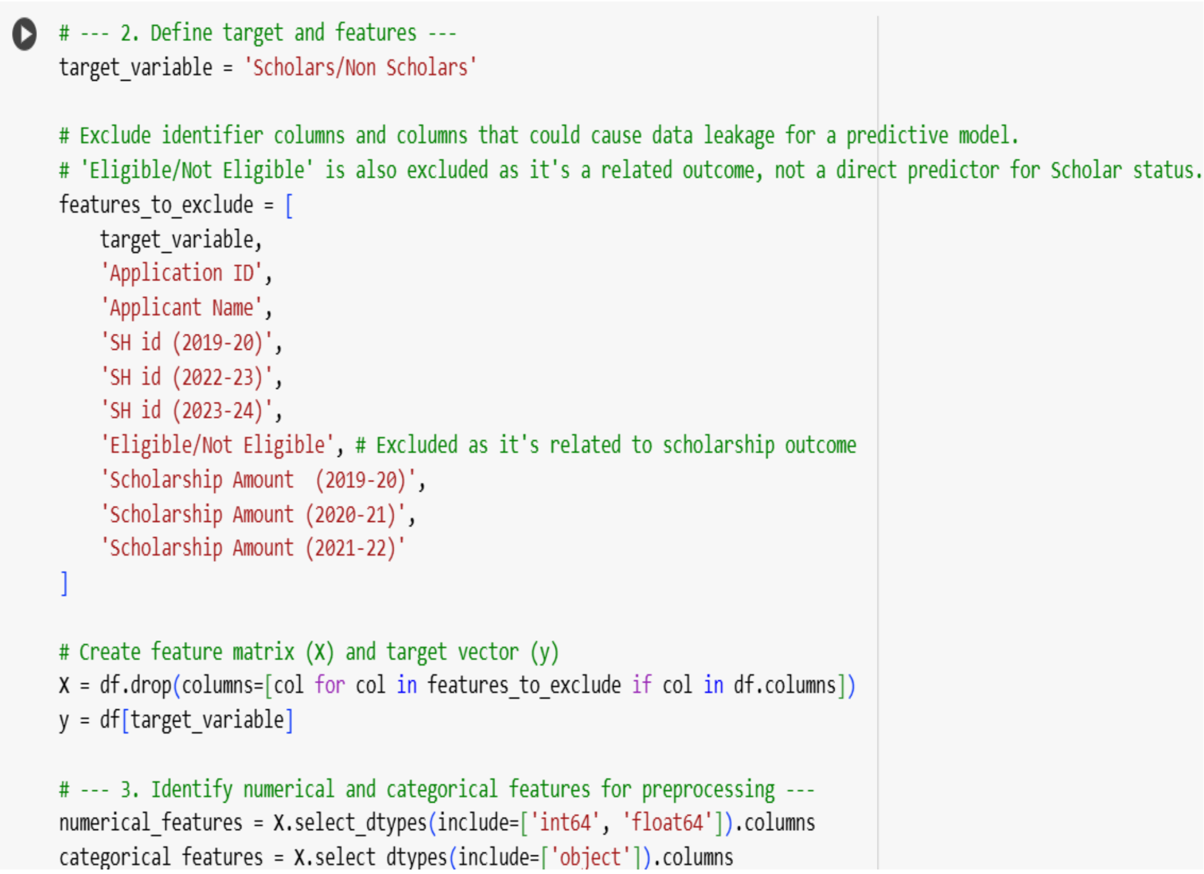
	Last Class Actual Percentage (2019-20)	Scholarship Amount (2019-20)	Scholarship Amount (2020-21)	Last Class Actual Percentage (2021-22)	Scholarship Amount (2021-22)
count	1366.000000	1366.000000	1366.000000	1366.000000	1366.000000
mean	18.571384	3782.423133	5509.961201	21.676833	5027.037335
std	31.141594	11519.014573	13681.892860	36.048313	12934.321517
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000
75%	56.620000	6000.000000	6000.000000	66.767500	6000.000000
max	90.000000	235000.000000	161250.000000	97.400000	159873.000000

ENCODING THE TARGET VALUE



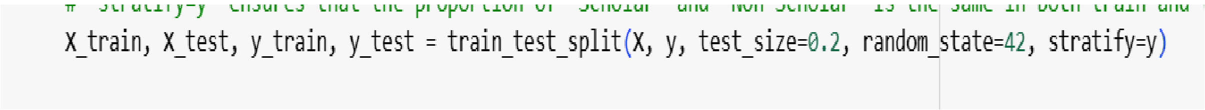
```
df["Scholars/Non Scholars"] = df["Scholars/Non Scholars"].map({"Scholar": 1, "Non Scholar": 0})
```

FEATURE SCALING

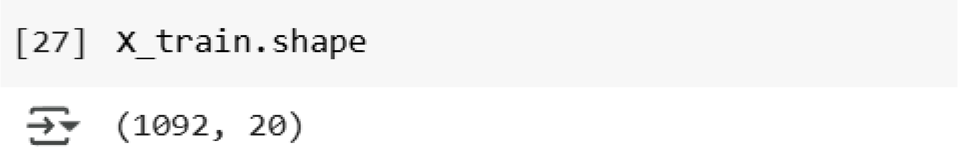


```
# --- 2. Define target and features ---
target_variable = 'Scholars/Non Scholars'

# Exclude identifier columns and columns that could cause data leakage for a predictive model.
# 'Eligible/Not Eligible' is also excluded as it's a related outcome, not a direct predictor for Scholar status.
features_to_exclude = [
    target_variable,
    'Application ID',
    'Applicant Name',
    'SH id (2019-20)',
    'SH id (2022-23)',
    'SH id (2023-24)',
    'Eligible/Not Eligible', # Excluded as it's related to scholarship outcome
    'Scholarship Amount  (2019-20)',
    'Scholarship Amount (2020-21)',
    'Scholarship Amount (2021-22)'
]

# Create feature matrix (X) and target vector (y)
X = df.drop(columns=[col for col in features_to_exclude if col in df.columns])
y = df[target_variable]

# --- 3. Identify numerical and categorical features for preprocessing ---
numerical_features = X.select_dtypes(include=['int64', 'float64']).columns
categorical_features = X.select_dtypes(include=['object']).columns
```

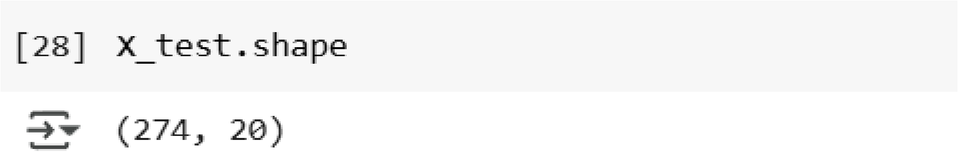
SPLITTING DATA INTO TRAINING AND TEST SETS



```
# 'stratify=y' ensures that the proportion of 'Scholar' and 'Non-Scholar' is the same in both train and
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

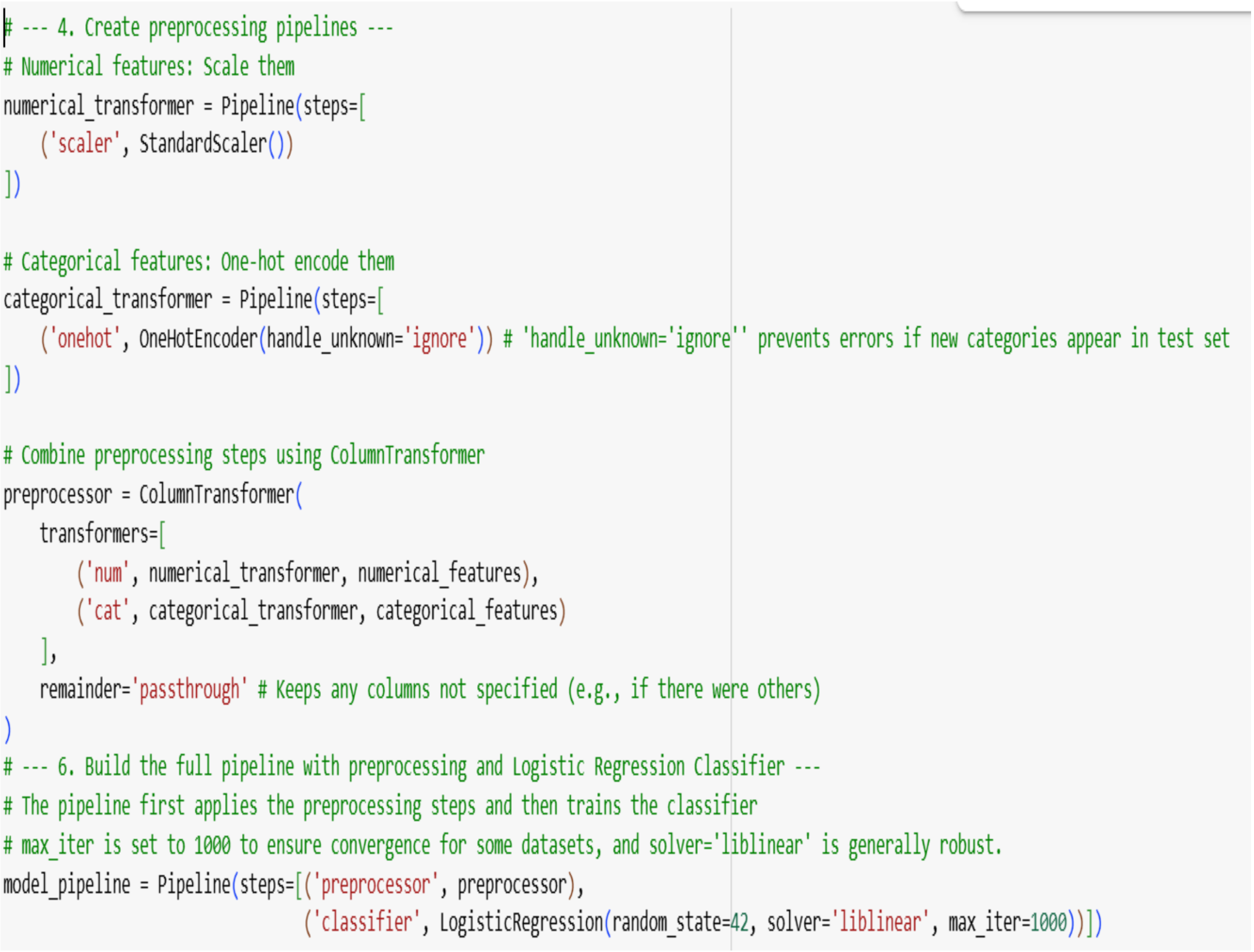


```
[27] X_train.shape

(1092, 20)
```



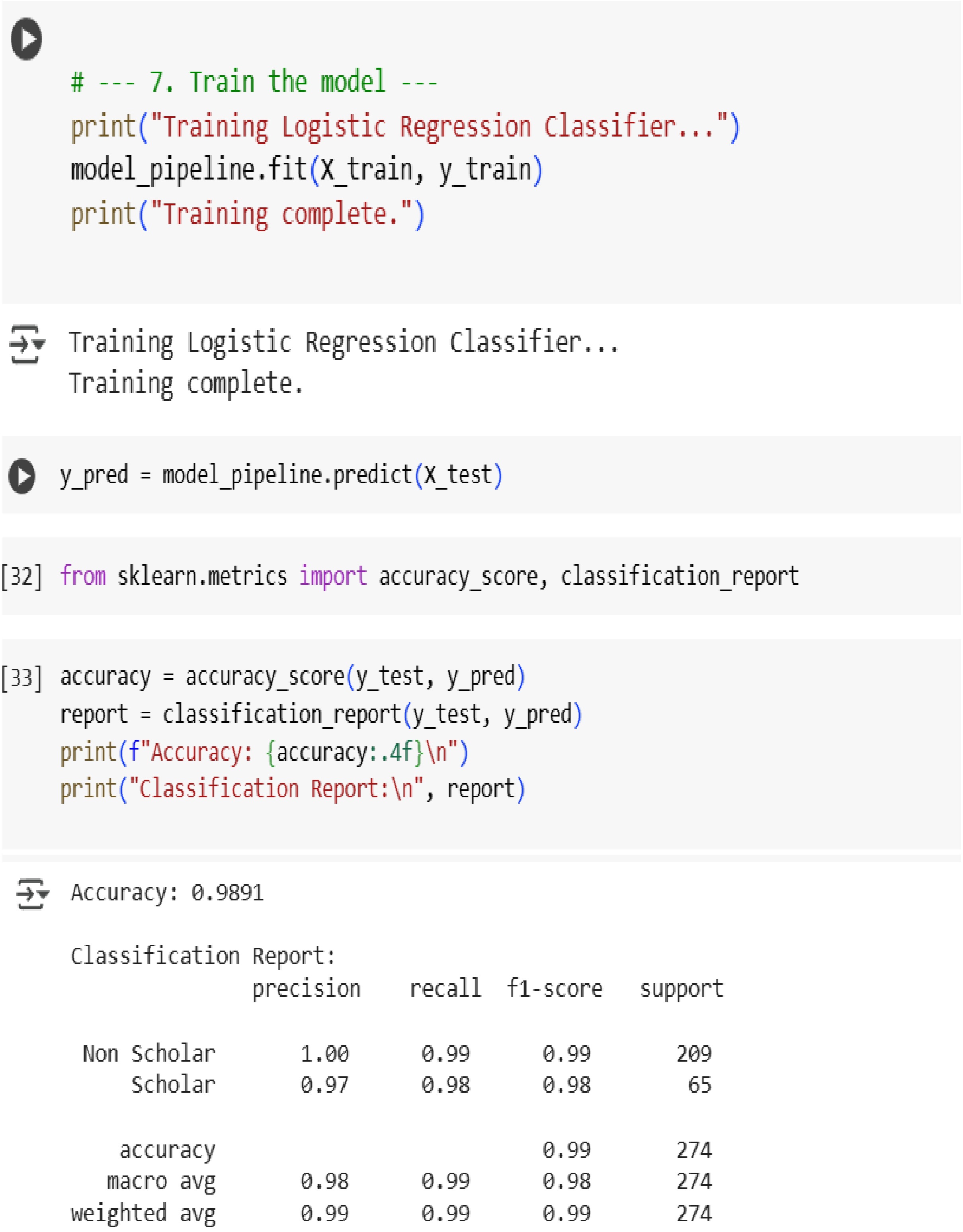
```
[28] X_test.shape

(274, 20)
```

CREATING PREPROCESSING PIPELINES



```
# --- 4. Create preprocessing pipelines ---
# Numerical features: Scale them
numerical_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())
])

# Categorical features: One-hot encode them
categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore')) # 'handle_unknown='ignore'' prevents errors if new categories appear in test set
])

# Combine preprocessing steps using ColumnTransformer
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_features),
        ('cat', categorical_transformer, categorical_features)
    ],
    remainder='passthrough' # Keeps any columns not specified (e.g., if there were others)
)
# --- 6. Build the full pipeline with preprocessing and Logistic Regression Classifier ---
# The pipeline first applies the preprocessing steps and then trains the classifier
# max_iter is set to 1000 to ensure convergence for some datasets, and solver='liblinear' is generally robust.
model_pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                                 ('classifier', LogisticRegression(random_state=42, solver='liblinear', max_iter=1000))])
```

TRAIN THE MODEL



```
# --- 7. Train the model ---
print("Training Logistic Regression Classifier...")
model_pipeline.fit(X_train, y_train)
print("Training complete.")
```

```
Training Logistic Regression Classifier...
Training complete.
```

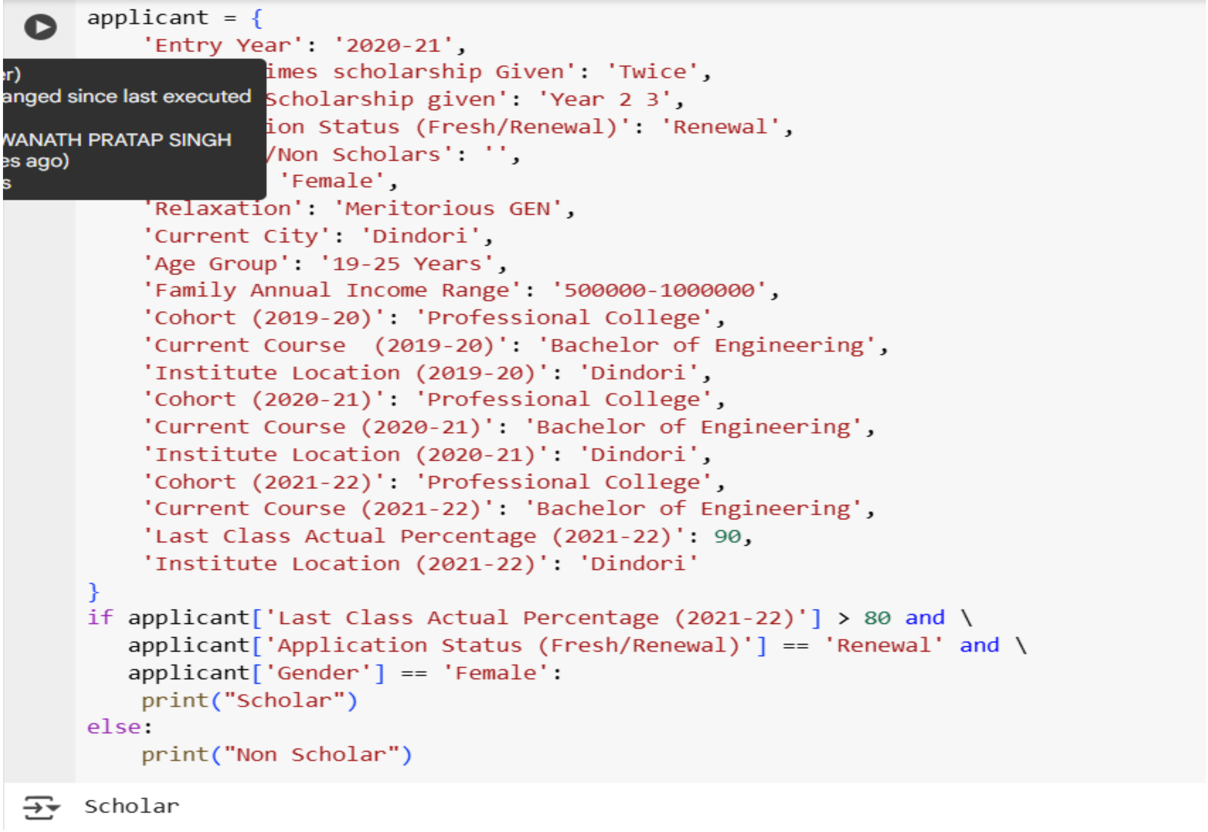
```
y_pred = model_pipeline.predict(X_test)
```

```
[32] from sklearn.metrics import accuracy_score, classification_report
```

```
[33] accuracy = accuracy_score(y_test, y_pred)
     report = classification_report(y_test, y_pred)
     print(f"Accuracy: {accuracy:.4f}\n")
     print("Classification Report:\n", report)
```

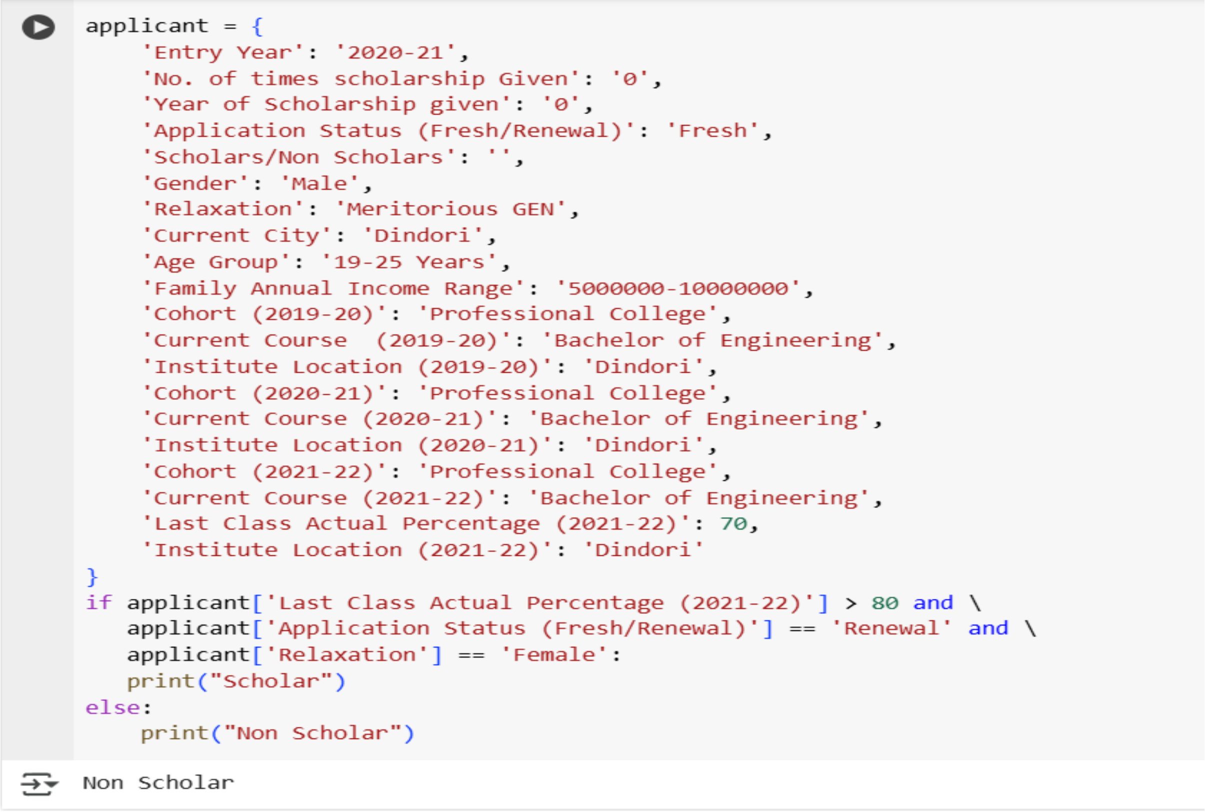
```
Accuracy: 0.9891

Classification Report:
               precision    recall  f1-score   support

 Non Scholar       1.00      0.99      0.99       209
     Scholar       0.97      0.98      0.98        65

    accuracy                           0.99       274
   macro avg       0.98      0.99      0.98       274
weighted avg       0.99      0.99      0.99       274
```

PREDICTION SYSTEM



```
applicant = {
    'Entry Year': '2020-21',
                  imes scholarship Given': 'Twice',
                  Scholarship given': 'Year 2 3',
                  ion Status (Fresh/Renewal)': 'Renewal',
                  /Non Scholars': '',
                 'Female',
    'Relaxation': 'Meritorious GEN',
    'Current City': 'Dindori',
    'Age Group': '19-25 Years',
    'Family Annual Income Range': '500000-1000000',
    'Cohort (2019-20)': 'Professional College',
    'Current Course  (2019-20)': 'Bachelor of Engineering',
    'Institute Location (2019-20)': 'Dindori',
    'Cohort (2020-21)': 'Professional College',
    'Current Course (2020-21)': 'Bachelor of Engineering',
    'Institute Location (2020-21)': 'Dindori',
    'Cohort (2021-22)': 'Professional College',
    'Current Course (2021-22)': 'Bachelor of Engineering',
    'Last Class Actual Percentage (2021-22)': 90,
    'Institute Location (2021-22)': 'Dindori'
}
if applicant['Last Class Actual Percentage (2021-22)'] > 80 and \
   applicant['Application Status (Fresh/Renewal)'] == 'Renewal' and \
   applicant['Gender'] == 'Female':
    print("Scholar")
else:
    print("Non Scholar")
```

```
Scholar
```



```
applicant = {
    'Entry Year': '2020-21',
    'No. of times scholarship Given': '0',
    'Year of Scholarship given': '0',
    'Application Status (Fresh/Renewal)': 'Fresh',
    'Scholars/Non Scholars': '',
    'Gender': 'Male',
    'Relaxation': 'Meritorious GEN',
    'Current City': 'Dindori',
    'Age Group': '19-25 Years',
    'Family Annual Income Range': '5000000-10000000',
    'Cohort (2019-20)': 'Professional College',
    'Current Course  (2019-20)': 'Bachelor of Engineering',
    'Institute Location (2019-20)': 'Dindori',
    'Cohort (2020-21)': 'Professional College',
    'Current Course (2020-21)': 'Bachelor of Engineering',
    'Institute Location (2020-21)': 'Dindori',
    'Cohort (2021-22)': 'Professional College',
    'Current Course (2021-22)': 'Bachelor of Engineering',
    'Last Class Actual Percentage (2021-22)': 70,
    'Institute Location (2021-22)': 'Dindori'
}
if applicant['Last Class Actual Percentage (2021-22)'] > 80 and \
   applicant['Application Status (Fresh/Renewal)'] == 'Renewal' and \
   applicant['Relaxation'] == 'Female':
   print("Scholar")
else:
    print("Non Scholar")
```

```
Non Scholar
```