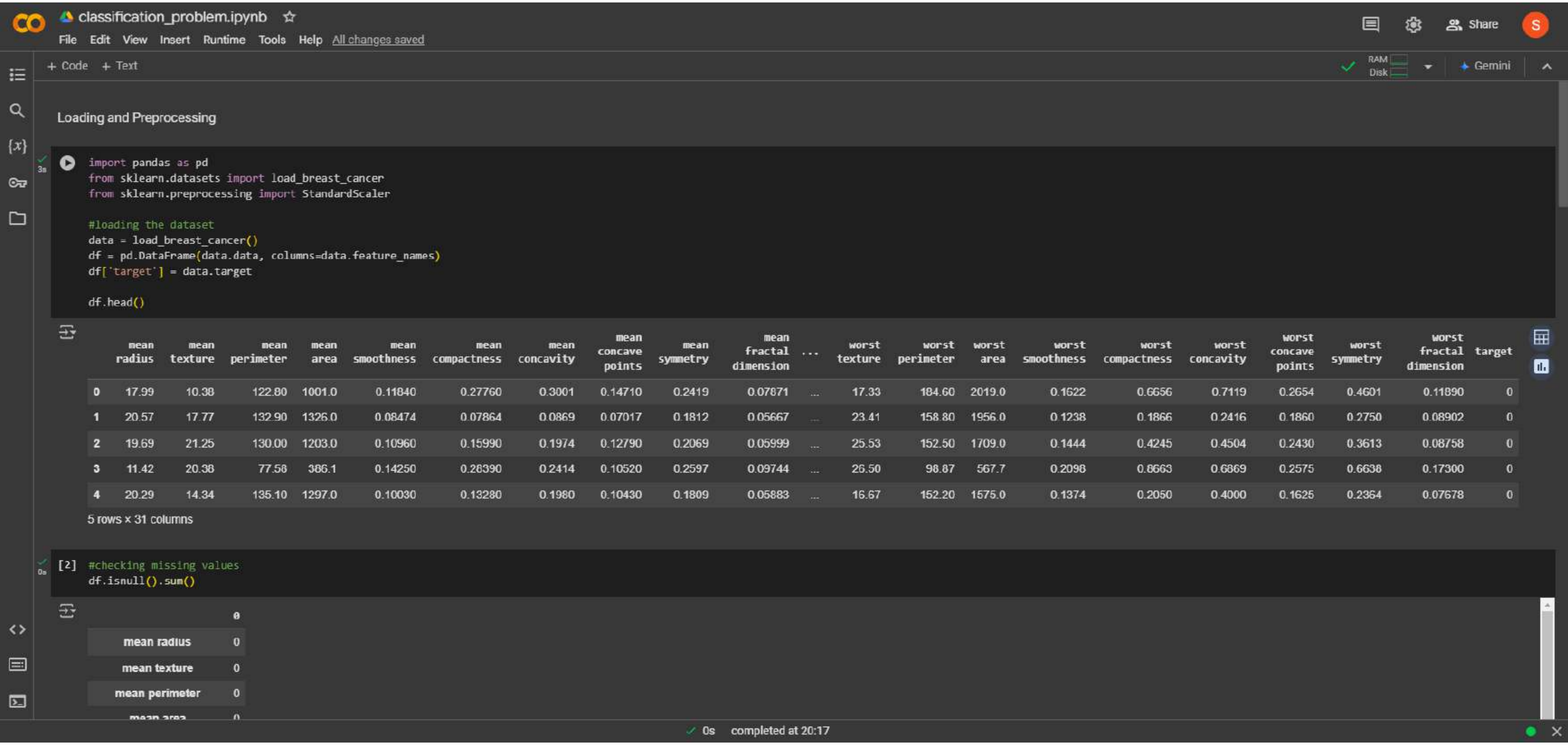

classification_problem.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Share
+ Code + Text
RAM
Disk
Gemini

Loading and Preprocessing

```python
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.preprocessing import StandardScaler

#loading the dataset
data = load_breast_cancer()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target

df.head()
```

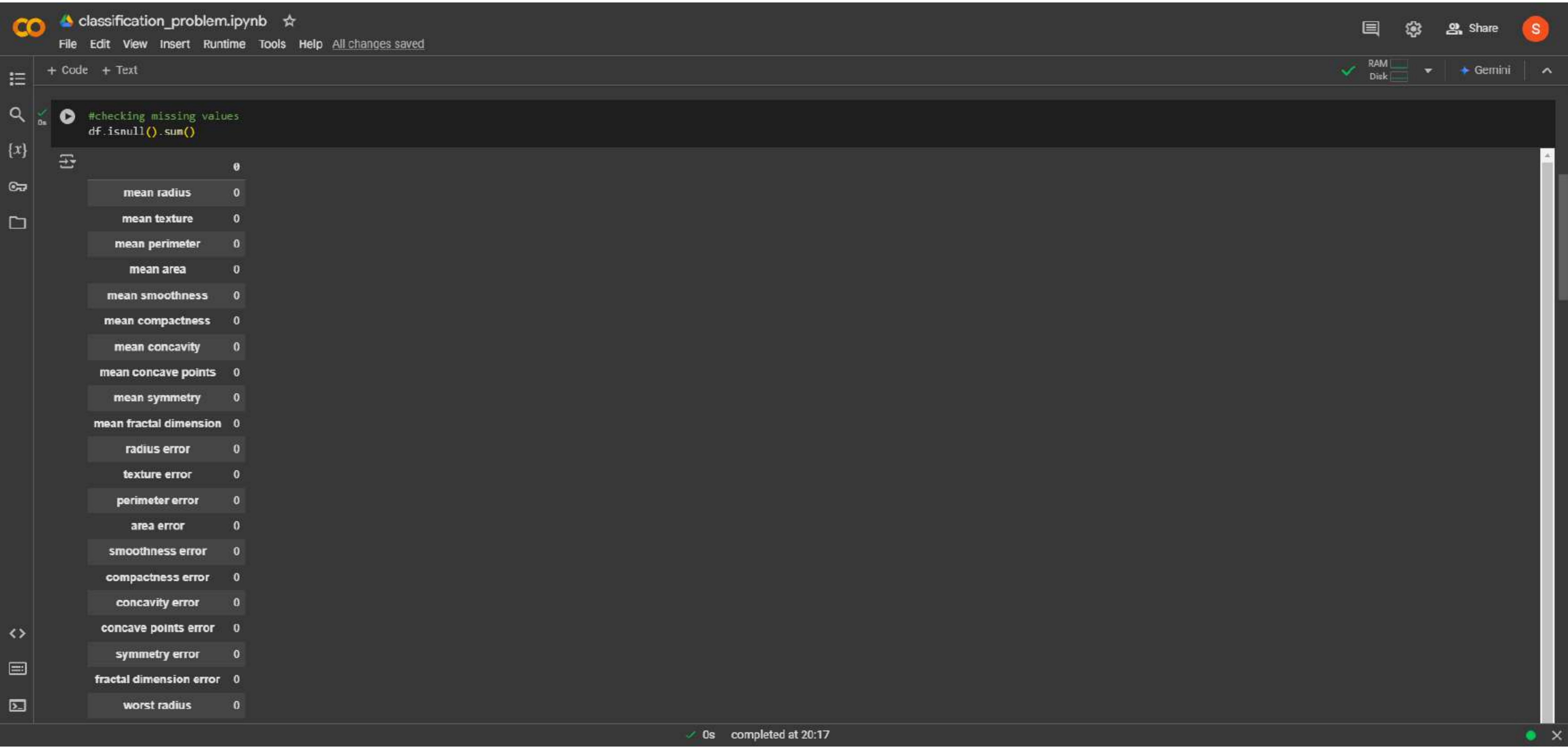
	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension	target
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.1622	0.6656	0.7119	0.2654	0.4601	0.11890	0
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.1238	0.1866	0.2416	0.1860	0.2750	0.08902	0
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.1444	0.4245	0.4504	0.2430	0.3613	0.08758	0
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87	567.7	0.2098	0.8663	0.6869	0.2575	0.6638	0.17300	0
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0	0.1374	0.2050	0.4000	0.1625	0.2364	0.07678	0

5 rows × 31 columns

```python
[2] #checking missing values
df.isnull().sum()
```

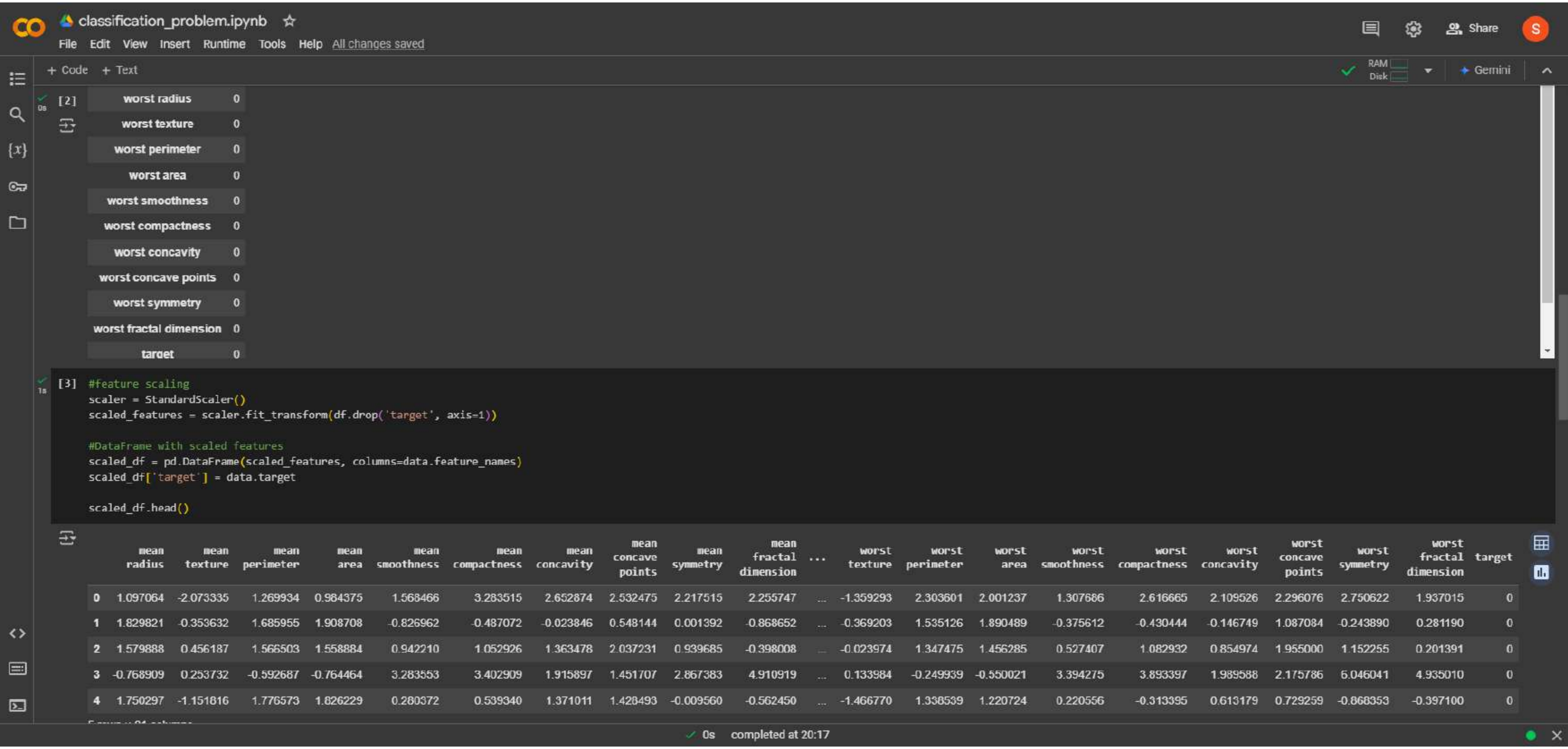
	0
mean radius	0
mean texture	0
mean perimeter	0
mean area	0

✓ 0s completed at 20:17


classification_problem.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Share
+ Code + Text
RAM
Disk
Gemini

```
#checking missing values
df.isnull().sum()
```

```
                          0
           mean radius    0
          mean texture    0
        mean perimeter    0
             mean area    0
       mean smoothness    0
      mean compactness    0
        mean concavity    0
   mean concave points    0
         mean symmetry    0
mean fractal dimension    0
          radius error    0
         texture error    0
       perimeter error    0
            area error    0
      smoothness error    0
     compactness error    0
       concavity error    0
  concave points error    0
        symmetry error    0
fractal dimension error   0
          worst radius    0
```


classification_problem.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Share
+ Code + Text
RAM
Disk
Gemini

[2]

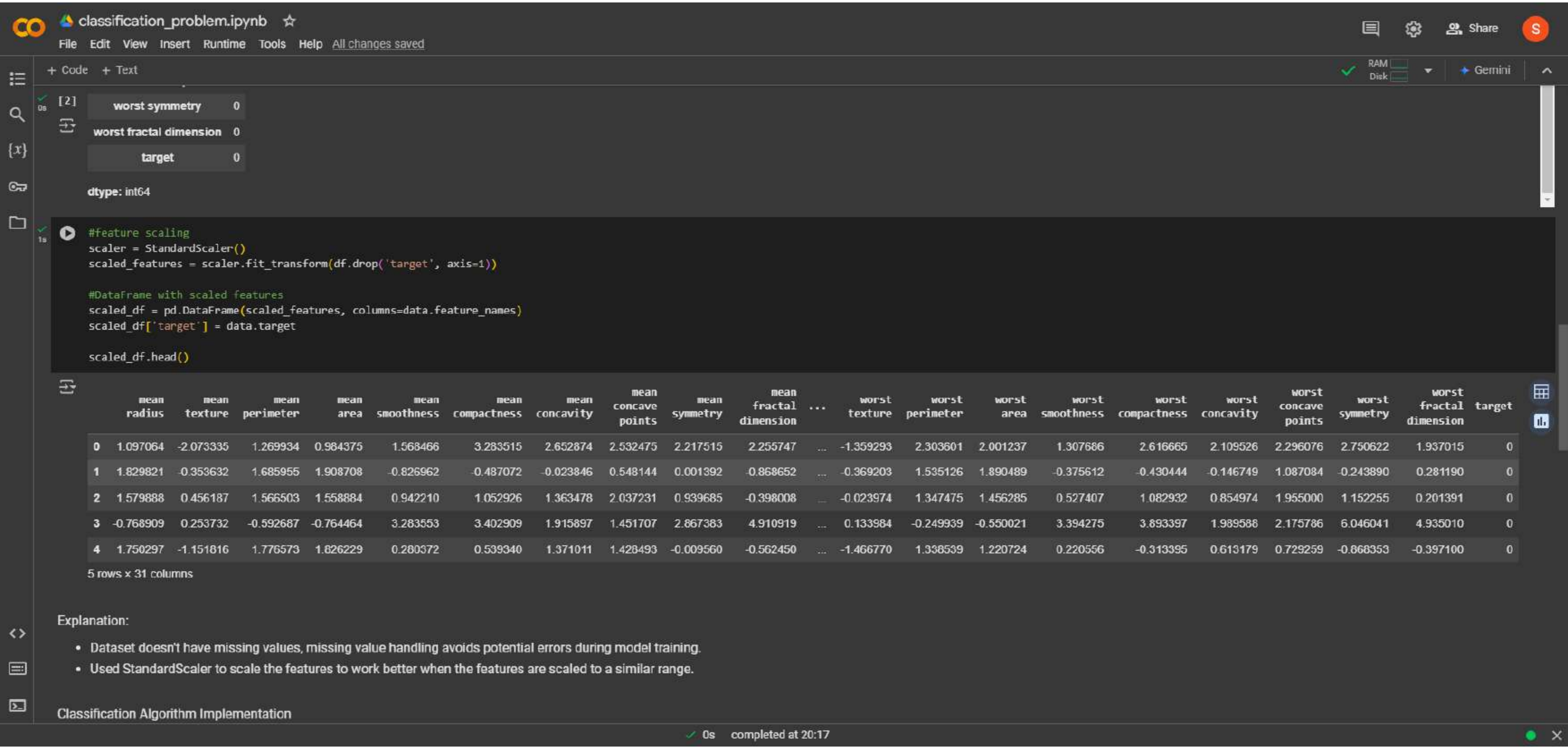
worst radius	0
worst texture	0
worst perimeter	0
worst area	0
worst smoothness	0
worst compactness	0
worst concavity	0
worst concave points	0
worst symmetry	0
worst fractal dimension	0
target	0

[3]

```
#feature scaling
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df.drop('target', axis=1))

#DataFrame with scaled features
scaled_df = pd.DataFrame(scaled_features, columns=data.feature_names)
scaled_df['target'] = data.target

scaled_df.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension	target
0	1.097064	-2.073335	1.269934	0.984375	1.568466	3.283515	2.652874	2.532475	2.217515	2.255747	...	-1.359293	2.303601	2.001237	1.307686	2.616665	2.109526	2.296076	2.750622	1.937015	0
1	1.829821	-0.353632	1.685955	1.908708	-0.826962	-0.487072	-0.023846	0.548144	0.001392	-0.868652	...	-0.369203	1.535126	1.890489	-0.375612	-0.430444	-0.146749	1.087084	-0.243890	0.281190	0
2	1.579888	0.456187	1.566503	1.558884	0.942210	1.052926	1.363478	2.037231	0.939685	-0.398008	...	-0.023974	1.347475	1.456285	0.527407	1.082932	0.854974	1.955000	1.152255	0.201391	0
3	-0.768909	0.253732	-0.592687	-0.764464	3.283553	3.402909	1.915897	1.451707	2.867383	4.910919	...	0.133984	-0.249939	-0.550021	3.394275	3.893397	1.989588	2.175786	6.046041	4.935010	0
4	1.750297	-1.151816	1.776573	1.826229	0.280372	0.539340	1.371011	1.428493	-0.009560	-0.562450	...	-1.466770	1.338539	1.220724	0.220556	-0.313395	0.613179	0.729259	-0.868353	-0.397100	0

✓ 0s completed at 20:17



classification_problem.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Share
+ Code + Text
Gemini

worst symmetry	0
worst fractal dimension	0
target	0

dtype: int64

```
#feature scaling
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df.drop('target', axis=1))

#DataFrame with scaled features
scaled_df = pd.DataFrame(scaled_features, columns=data.feature_names)
scaled_df['target'] = data.target

scaled_df.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension	target
0	1.097064	-2.073335	1.269934	0.984375	1.568466	3.283515	2.652874	2.532475	2.217515	2.255747	...	-1.359293	2.303601	2.001237	1.307686	2.616665	2.109526	2.296076	2.750622	1.937015	0
1	1.829821	-0.353632	1.685955	1.908708	-0.826962	-0.487072	-0.023846	0.548144	0.001392	-0.868652	...	-0.369203	1.535126	1.890489	-0.375612	-0.430444	-0.146749	1.087084	-0.243890	0.281190	0
2	1.579888	0.456187	1.566503	1.558884	0.942210	1.052926	1.363478	2.037231	0.939685	-0.398008	...	-0.023974	1.347475	1.456285	0.527407	1.082932	0.854974	1.955000	1.152255	0.201391	0
3	-0.768909	0.253732	-0.592687	-0.764464	3.283553	3.402909	1.915897	1.451707	2.867383	4.910919	...	0.133984	-0.249939	-0.550021	3.394275	3.893397	1.989588	2.175786	6.046041	4.935010	0
4	1.750297	-1.151816	1.776573	1.826229	0.280372	0.539340	1.371011	1.428493	-0.009560	-0.562450	...	-1.466770	1.338539	1.220724	0.220556	-0.313395	0.613179	0.729259	-0.868353	-0.397100	0

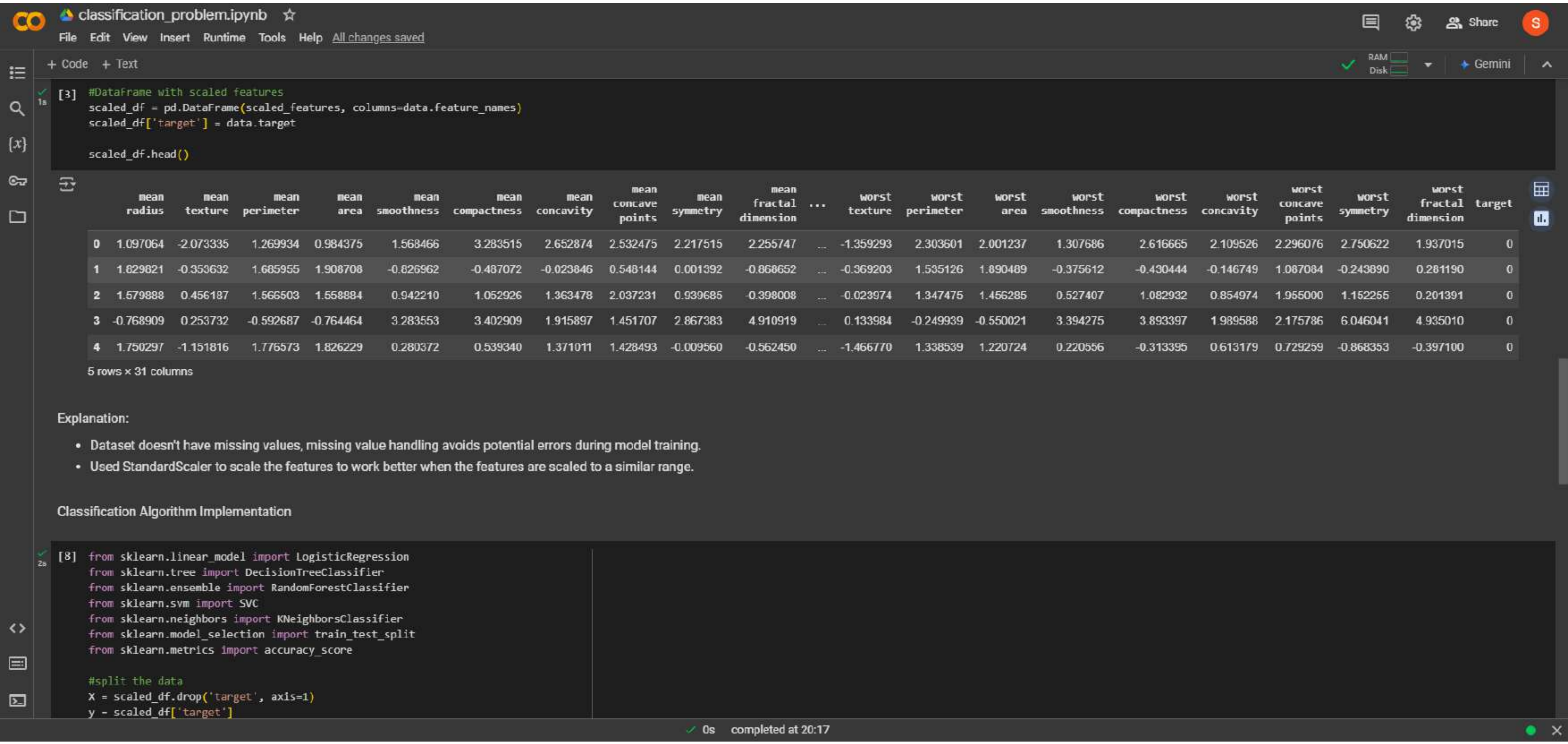
5 rows × 31 columns

Explanation:

- Dataset doesn't have missing values, missing value handling avoids potential errors during model training.
- Used StandardScaler to scale the features to work better when the features are scaled to a similar range.

Classification Algorithm Implementation

```
✓ 0s completed at 20:17
```


classification_problem.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Share
+ Code + Text
RAM Disk
Gemini

```
[3] #DataFrame with scaled features
    scaled_df = pd.DataFrame(scaled_features, columns=data.feature_names)
    scaled_df['target'] = data.target

    scaled_df.head()
```

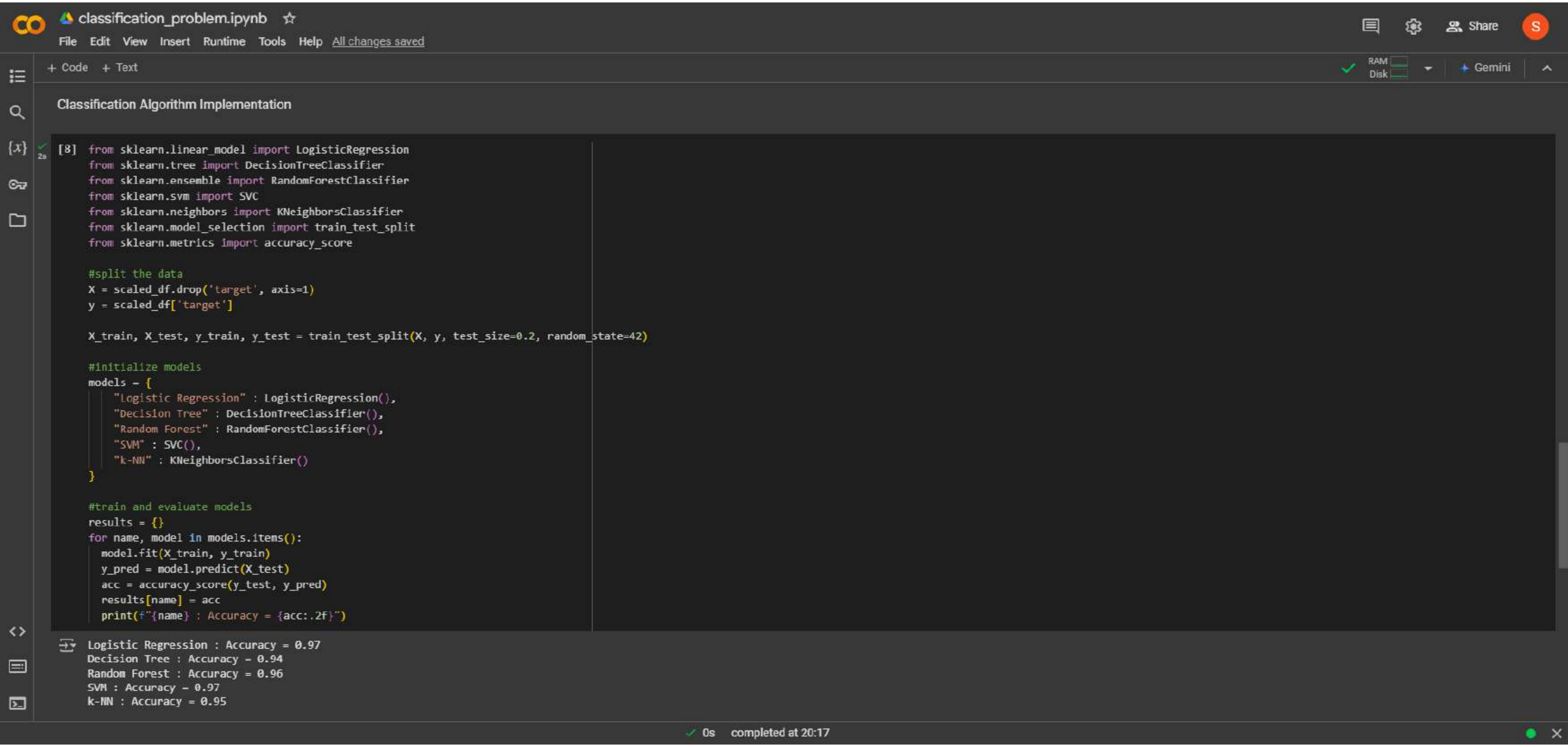
```
       mean     mean       mean      mean        mean         mean       mean     mean     mean      mean  ...     worst      worst      worst       worst        worst       worst     worst     worst     worst  target
     radius  texture  perimeter      area  smoothness  compactness  concavity  concave  symmetry  fractal  ...   texture  perimeter      area  smoothness  compactness   concavity  concave  symmetry   fractal
                                                                                points            dimension                                                                          points             dimension
0  1.097064 -2.073335   1.269934  0.984375    1.568466     3.283515   2.652874  2.532475  2.217515  2.255747  ...  -1.359293   2.303601   2.001237    1.307686     2.616665   2.109526  2.296076  2.750622   1.937015      0
1  1.829821 -0.353632   1.685955  1.908708   -0.826962    -0.487072  -0.023846  0.548144  0.001392 -0.868652  ...  -0.369203   1.535126   1.890489   -0.375612    -0.430444  -0.146749  1.087084 -0.243890   0.281190      0
2  1.579888  0.456187   1.566503  1.558884    0.942210     1.052926   1.363478  2.037231  0.939685 -0.398008  ...  -0.023974   1.347475   1.456285    0.527407     1.082932   0.854974  1.955000  1.152255   0.201391      0
3 -0.768909  0.253732  -0.592687 -0.764464    3.283553     3.402909   1.915897  1.451707  2.867383  4.910919  ...   0.133984  -0.249939  -0.550021    3.394275     3.893397   1.989588  2.175786  6.046041   4.935010      0
4  1.750297 -1.151816   1.776573  1.826229    0.280372     0.539340   1.371011  1.428493 -0.009560 -0.562450  ...  -1.466770   1.338539   1.220724    0.220556    -0.313395   0.613179  0.729259 -0.868353  -0.397100      0

5 rows × 31 columns
```

Explanation:

- Dataset doesn't have missing values, missing value handling avoids potential errors during model training.
- Used StandardScaler to scale the features to work better when the features are scaled to a similar range.

Classification Algorithm Implementation

```
[8] from sklearn.linear_model import LogisticRegression
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.svm import SVC
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import accuracy_score

    #split the data
    X = scaled_df.drop('target', axis=1)
    y - scaled_df['target']
```

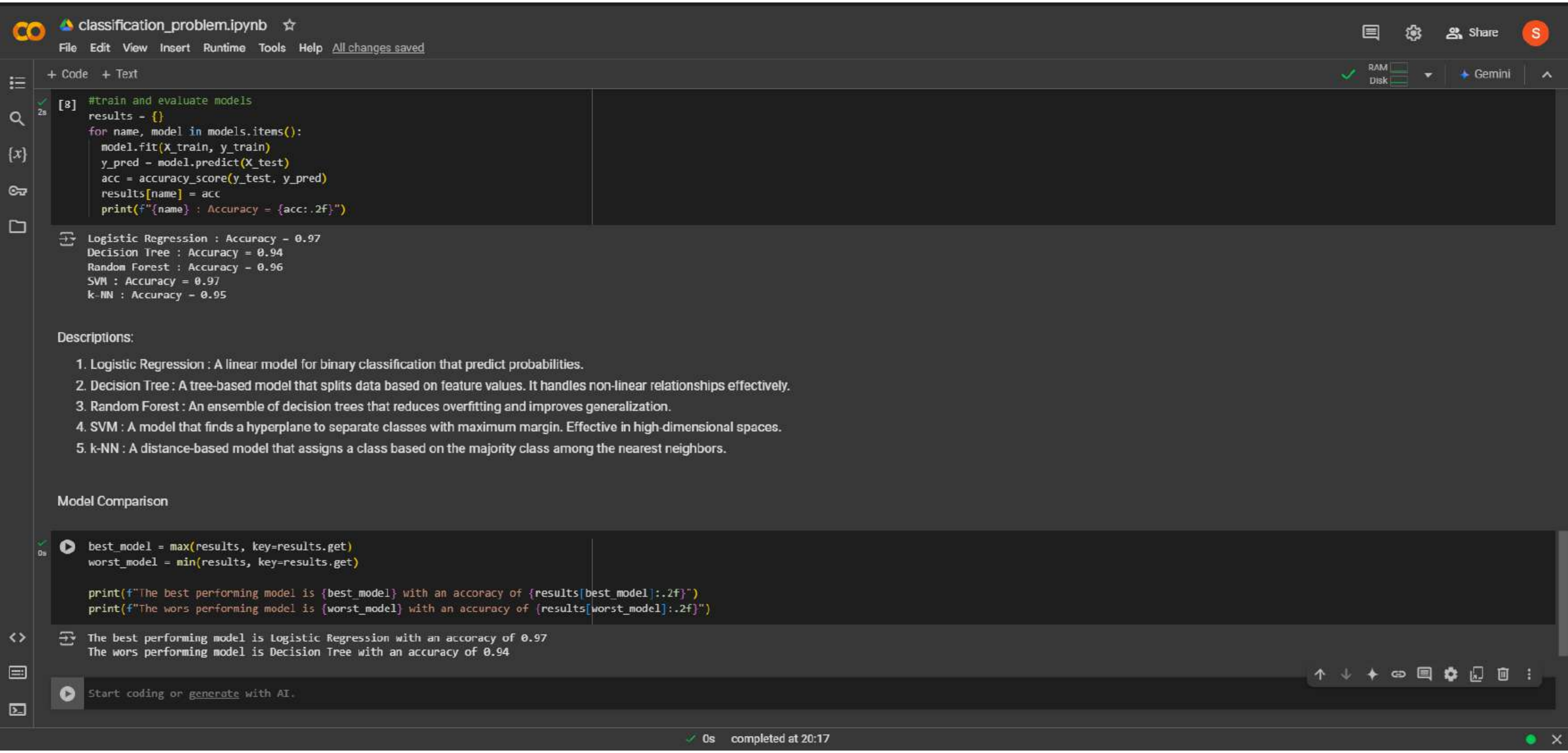

classification_problem.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Share
+ Code + Text
RAM Disk
Gemini

Classification Algorithm Implementation

```
[8] from sklearn.linear_model import LogisticRegression
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.svm import SVC
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import accuracy_score

    #split the data
    X = scaled_df.drop('target', axis=1)
    y = scaled_df['target']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    #initialize models
    models = {
        "Logistic Regression" : LogisticRegression(),
        "Decision Tree" : DecisionTreeClassifier(),
        "Random Forest" : RandomForestClassifier(),
        "SVM" : SVC(),
        "k-NN" : KNeighborsClassifier()
    }

    #train and evaluate models
    results = {}
    for name, model in models.items():
      model.fit(X_train, y_train)
      y_pred = model.predict(X_test)
      acc = accuracy_score(y_test, y_pred)
      results[name] = acc
      print(f"{name} : Accuracy = {acc:.2f}")
```

```
Logistic Regression : Accuracy = 0.97
Decision Tree : Accuracy = 0.94
Random Forest : Accuracy = 0.96
SVM : Accuracy = 0.97
k-NN : Accuracy = 0.95
```

✓ 0s completed at 20:17


classification_problem.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Share
+ Code + Text
RAM Disk
Gemini

```
[8]
#train and evaluate models
results = {}
for name, model in models.items():
  model.fit(X_train, y_train)
  y_pred = model.predict(X_test)
  acc = accuracy_score(y_test, y_pred)
  results[name] = acc
  print(f"{name} : Accuracy = {acc:.2f}")
```

```
Logistic Regression : Accuracy = 0.97
Decision Tree : Accuracy = 0.94
Random Forest : Accuracy = 0.96
SVM : Accuracy = 0.97
k-NN : Accuracy = 0.95
```

Descriptions:

1. Logistic Regression : A linear model for binary classification that predict probabilities.
2. Decision Tree : A tree-based model that splits data based on feature values. It handles non-linear relationships effectively.
3. Random Forest : An ensemble of decision trees that reduces overfitting and improves generalization.
4. SVM : A model that finds a hyperplane to separate classes with maximum margin. Effective in high-dimensional spaces.
5. k-NN : A distance-based model that assigns a class based on the majority class among the nearest neighbors.

Model Comparison

```
best_model = max(results, key=results.get)
worst_model = min(results, key=results.get)

print(f"The best performing model is {best_model} with an accoracy of {results[best_model]:.2f}")
print(f"The wors performing model is {worst_model} with an accuracy of {results[worst_model]:.2f}")
```

```
The best performing model is Logistic Regression with an accoracy of 0.97
The wors performing model is Decision Tree with an accuracy of 0.94
```

```
Start coding or generate with AI.
```