Assignment 2

1. (10 pts.) Clean up the data set. There are tuples in the data set in which there are missing values. Based on what you have learned in the course, propose and implement a method to clean up the data set. Note that the data set includes adult.data and adult.test both files.

There are some tuples in the data set that have missing values and are represented by '?'

Approach 1: These values are replaced with NaN, and then NaN values are dropped. The sizes of both the dataset after cleaning matches what's mentioned in the adult.names file.

df_train before cleaning = 3261 rows x 15 columns

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32556	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38	United-States	<=50K
32557	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40	United-States	>50K
32558	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40	United-States	<=50K
32559	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	20	United-States	<=50K
32560	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0	40	United-States	>50K

32561 rows × 15 columns

df_train after cleaning = 30162 rows x 15 columns

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32556	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38	United-States	<=50K
32557	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40	United-States	>50K
32558	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40	United-States	<=50K
32559	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	20	United-States	<=50K
32560	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0	40	United-States	>50K

30162 rows × 15 columns

df_test before cleaning = 16281 rows x 15 columns

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States	<=50K
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States	<=50K
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States	>50K
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States	>50K
4	18	?	103497	Some-college	10	Never-married	?	Own-child	White	Female	0	0	30	United-States	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
16276	39	Private	215419	Bachelors	13	Divorced	Prof-specialty	Not-in-family	White	Female	0	0	36	United-States	<=50K
16277	64	?	321403	HS-grad	9	Widowed	?	Other-relative	Black	Male	0	0	40	United-States	<=50K
16278	38	Private	374983	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	0	50	United-States	<=50K
16279	44	Private	83891	Bachelors	13	Divorced	Adm-clerical	Own-child	Asian-Pac-Islander	Male	5455	0	40	United-States	<=50K
16280	35	Self-emp-inc	182148	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	60	United-States	>50K

16281 rows × 15 columns

df_test after cleaning = 15060 rows x 15 columns

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States	<=50K
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States	<=50K
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States	>50K
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States	>50K
5	34	Private	198693	10th	6	Never-married	Other-service	Not-in-family	White	Male	0	0	30	United-States	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
16275	33	Private	245211	Bachelors	13	Never-married	Prof-specialty	Own-child	White	Male	0	0	40	United-States	<=50K
16276	39	Private	215419	Bachelors	13	Divorced	Prof-specialty	Not-in-family	White	Female	0	0	36	United-States	<=50K
16278	38	Private	374983	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	0	50	United-States	<=50K
16279	44	Private	83891	Bachelors	13	Divorced	Adm-clerical	Own-child	Asian-Pac-Islander	Male	5455	0	40	United-States	<=50K
16280	35	Self-emp-inc	182148	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	60	United-States	>50K

15060 rows × 15 columns

Approach 2: Instead of dropping rows with NaN values, a more effective strategy is to impute the missing values with the most frequent value in each column, which is the mode. This approach ensures that the imputed values are representative of the entire dataset. Following this methodology, I have updated both the train and test datasets accordingly.

```
question_mark_count = (df_train == ' ?').sum()
print(question_mark_count)
```

```
age                0
workclass       1836
fnlwgt             0
education          0
education.num      0
marital.status     0
occupation      1843
relationship       0
race               0
sex                0
capital.gain       0
capital.loss       0
hours.per.week     0
native.country   583
income             0
dtype: int64
```

Here it can be seen how the missing values are replaced. This is now the cleaned training dataset. Similarly after performing similar preprocessing for the test dataset, both datasets are ready to be trained and tested upon.

```
[ ] df_train['occupation'].describe()
```

```
count              30718
unique                14
top        Prof-specialty
freq                4140
Name: occupation, dtype: object
```

```
[ ] #Since mode is Prof-specialty, replacing null values with it

    df_train['occupation'] = df_train['occupation'].fillna('Prof-specialty')
```

```
df_train['workclass'].describe()
```

```
count        30725
unique           8
top        Private
freq         22696
Name: workclass, dtype: object
```

```
[ ] df_train['workclass'] = df_train['workclass'].fillna('Private')
```

```
[ ] df_train['native.country'].describe()
```

```
count              31978
unique                41
top        United-States
freq               29170
Name: native.country, dtype: object
```

```
[ ] df_train['native.country'] = df_train['native.country'].fillna('United-States')
```

2. (30 pts.) Implement a classification method either you have learned from this course or from the literature. Use adult.data as the training dataset and use adult.test as the evaluation dataset to report the classification error rate.

1. **K nearest neighbor:**

   training accuracy: 89.75154325727097
   training error: 10.248456742729033
   testing accuracy: 82.18168417173392
   testing error: 17.81831582826608

2. **Random Forest:**

   training accuracy: 99.96314609502164
   training error: 0.03685390497835783
   testing accuracy: 85.35716479331737
   testing error: 14.642835206682633

2. **Logistic regression:**

   training accuracy: 82.49132397653635
   training error: 17.50867602346365
   testing accuracy: 82.53178551686014
   testing error: 17.46821448313986

3. **Gaussian Naive Bayes:**

   training accuracy: 80.4
   training error: 19.599999999999994
   testing accuracy: 80.51102512130704
   testing error: 19.488974878692957

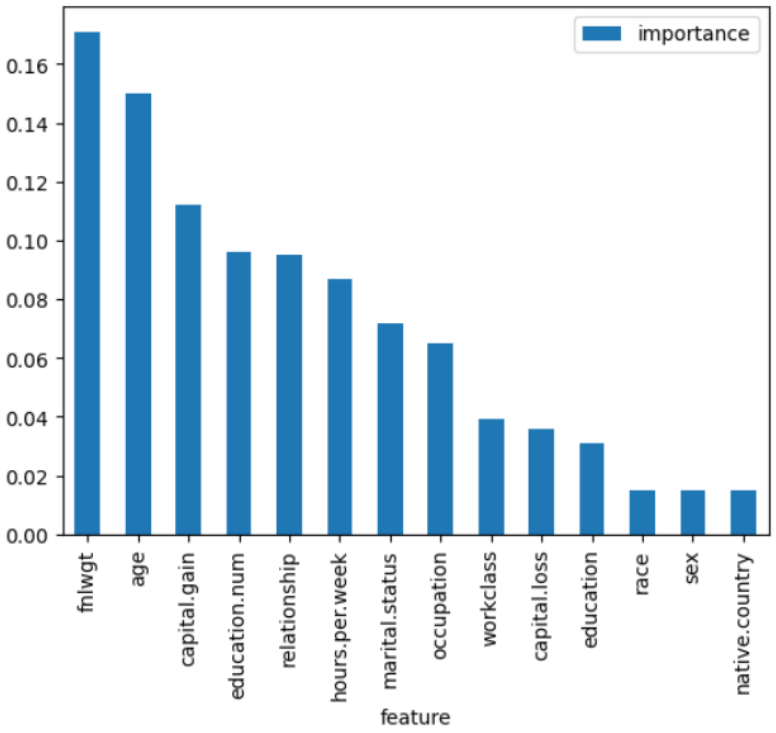
4. **Linear SVC:**

   training accuracy: 82.47
   training error: 17.53
   testing accuracy: 82.61163319206437
   testing error: 17.388366807935626

   I will select KNN and random forest to do further analyses in later part of the assignment

   After implementing random forest, I tried to see what features are the most important ones and have the most contribution to the training of the data.

The following graph shows the result.



I dropped the non significant features which are workclass, capital loss, education, race sex and native country and ran the random forest on it again. The result did not show any significant change and are recorded as follows

training accuracy: 98.66404594453488
training error: 1.3359540554651232
testing accuracy: 83.62508445427184
testing error: 16.374915545728157

No significant improvement in accuracy. It drops a little bit.

3. (10 pts.) Compare your classification error rate with the reported error rates in adult.names for the most well-known classification methods, and analyze the differences. (10 pts.) Suggest any improvements if there is a difference.

Information about the classifiers mentioned in adult.names file

C4.5: A decision tree algorithm used for classification that creates a tree by repeatedly splitting subsets of the data using the feature that provides the

best separation based on entropy or information gain. It has an accuracy of 84.46%, which means the error rate is 15.54%.

Naive-Bayes: A probabilistic classifier that applies Bayes' theorem with strong (naive) independence assumptions between the features. It is particularly known for its simplicity and efficiency. The accuracy for Naive-Bayes is given as 83.88%, thus the error rate is 16.12%.

NBTree: A hybrid model that combines Naive-Bayes and decision trees, hoping to balance the simplicity of Naive-Bayes with the decision-making structure of trees. NBTree outperforms both on this dataset with an accuracy of 85.90% and an error rate of 14.10%.

C4.5-auto: This might be an automated or enhanced version of C4.5 that could include optimizations like automatic feature selection or parameter tuning. The error rate for C4.5-auto is listed as 14.46%.

C4.5 rules: This takes the decision tree produced by C4.5 and converts it into a set of if-then rules. This approach can sometimes improve the interpretability and performance of the model. It has an error rate of 14.94%.

Voted ID3 (0.6) and Voted ID3 (0.8): These are likely variations of the ID3 algorithm, which is an earlier version of the decision tree algorithm leading up to C4.5. The numbers (0.6 and 0.8) might refer to a parameter for voting or confidence thresholds. The error rates for these are 15.64% and 16.47%, respectively.

Models implemented in part b

K-Nearest Neighbors (KNN):

Testing Error Rate: 17.82%

Analysis: The KNN model shows a higher error rate compared to most of the well-known methods listed in the adult.names file. Notably, it underperforms compared to models such as C4.5, NBTree, and FSS Naive Bayes.

Suggestion for Improvement: To enhance the KNN model's performance, we can consider tuning the number of neighbors (k) and exploring different distance metrics. Additionally, feature scaling and selection might further optimize the model's accuracy.

Random Forest:

Testing Error Rate: 15.35%

Analysis: The Random Forest model demonstrates a competitive error rate, closely aligning with the performance of C4.5 and OC1, and outperforming methods such as Naive-Bayes and Nearest-neighbor. This indicates that the Random Forest model is a strong contender for this classification task.

Suggestion for Improvement: To further improve the Random Forest model, increasing the number of trees and fine-tuning other hyperparameters could be beneficial. Additionally, exploring feature engineering techniques may enhance the model's predictive power.

Conclusion:

The Random Forest model exhibits promising results, with an error rate that is competitive with several well-known classification methods. While the KNN model shows a higher error rate, there is potential for improvement through hyperparameter tuning and feature optimization. Overall, the analysis suggests that with further refinement, both models could achieve improved performance on the Adult dataset.

4. (20 pts.) Randomly sample the training data set adult.data X% to obtain a new, down-sampled training data set; then use the down-sampled training dataset to train the classifier from (2) above and use the same evaluation data set adult.test to record the error rate. Repeat the whole process (i.e., randomly sample training data set --- train the classifier --- compute the error rate) five times and report the mean and the deviation for the error rate for each X when X is taken from 50, 60, 70, 80, and 90. Give an analysis on the reported results. Please note that when you down sample a training data set, you need to make sure that you randomly down sample data samples from each class with the given parameter of the sampling rate, i.e., X%.

KNN Downsampling

The original training dataset was randomly down sampled to create new training subsets representing 50%, 60%, 70%, 80%, and 90% of the original data volume. The down sampling process ensured that the proportional representation of each class was maintained

Results:

The error rates obtained for each downsampling percentage were as follows:

- 50%: 17.82%
- 60%: 18.11%
- 70%: 17.92%
- 80%: 17.89%
- 90%: 17.66%

The mean error rate across all experiments was 17.9%, with a standard deviation of 0.1%. The results suggest a negligible impact of training data volume on the classifier's error rate.

The minimal variation in error rates across different training data volumes indicates that the KNN classifier's performance is robust with respect to the size of the training dataset. The low standard deviation of 0.1% across the downsampling percentages reinforces the model's consistency. It is noteworthy that reducing the training data by up to 50% does not significantly degrade the classifier's performance.

The KNN classifier shows a strong resilience to the reduction of training data size, which implies that the full dataset may contain redundant information and that a smaller subset possesses sufficient predictive power. This robustness to downsampling can be particularly advantageous when computational efficiency is a concern.

Random Forest Downsampling

The error rates achieved after downsampling to the specified percentages were as follows:

50%: 15.24%

60%: 15.32%

70%: 15.23%

80%: 15.21%

90%: 15.24%

The mean error rate across these percentages was 15.3%, with a negligible standard deviation, indicating little to no variation in performance with the

reduction in training data size.

The consistency in error rates suggests that the Random Forest classifier maintains its predictive accuracy even when the available training data is halved. The lack of significant deviation implies that the model is robust and does not require the full volume of data to make accurate predictions.

The findings demonstrate the efficiency of Random Forest in utilizing smaller datasets to achieve results comparable to training on the full dataset. This could suggest potential for computational savings with minimal impact on model accuracy.

5. It is observed that all the classic classifiers are far from being perfect. Propose a solution that uses one classifier and the given training data set adult.data, if we use the same evaluation data set adult.test, we can beat all the classic classifiers reported in adult.names.

XGBoost (eXtreme Gradient Boosting) is a popular and efficient implementation of the gradient boosting framework. It is known for its speed and performance, and is widely used in machine learning.

When XGBoost was used to train and test this was the result:

The accuracy: 86.0082304526749
Testing error of model: 13.991769547325106

XGBoost has several features that contribute to its effectiveness:

1. **Gradient Boosting Framework:** XGBoost builds models sequentially to correct errors made by previous models, improving overall performance.
2. **Regularization:** XGBoost includes L1 and L2 regularization terms in the objective function, which helps to prevent overfitting and improve generalization.
3. **Handling Missing Values:** XGBoost can automatically handle missing values, reducing the need for extensive data preprocessing.
4. **Tree Pruning:** XGBoost uses a depth-first approach and prunes trees using the max_depth parameter, resulting in more efficient and optimized trees.

5. **Parallel and Distributed Computing:** XGBoost supports parallel and distributed computing, making it highly efficient and scalable for large datasets.

The higher accuracy achieved by XGBoost can be attributed to its ability to model complex relationships in the data through gradient boosting, regularization, and efficient tree pruning. These features likely contributed to its superior performance compared to classic classifiers.