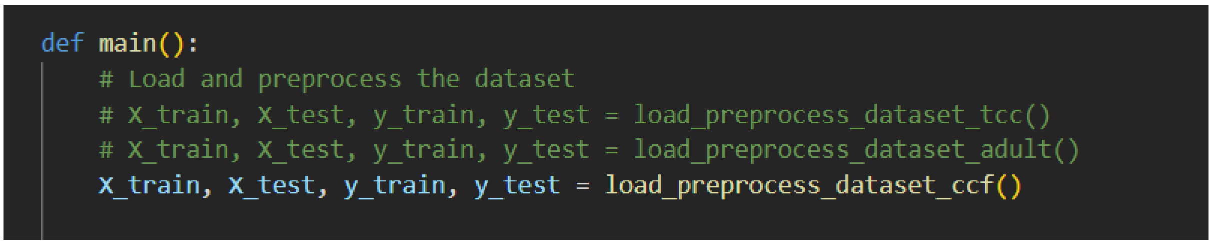
Offline 2 : Logistic Regression

Riad Ahmed Anonto
Student ID : 1905050

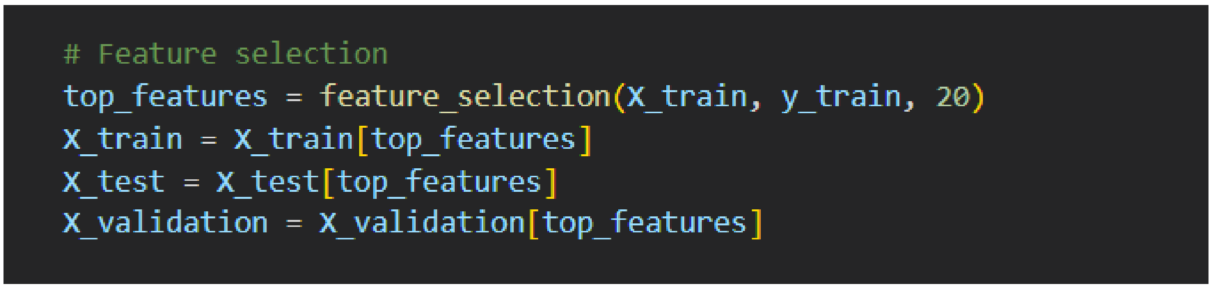
Steps to Run

1. Run the python code as it is. Uncomment the line for which dataset you want to run the code.



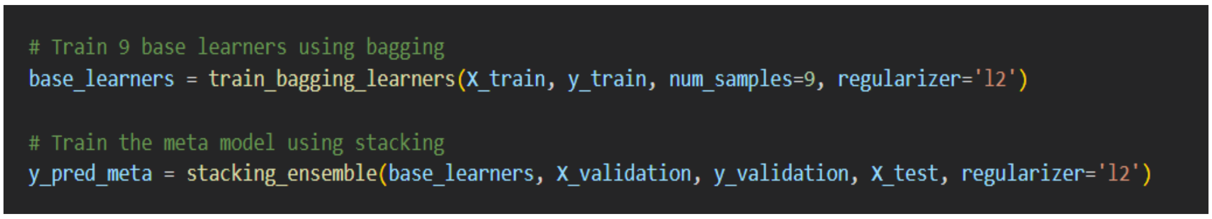
```python
def main():
    # Load and preprocess the dataset
    # X_train, X_test, y_train, y_test = load_preprocess_dataset_tcc()
    # X_train, X_test, y_train, y_test = load_preprocess_dataset_adult()
    X_train, X_test, y_train, y_test = load_preprocess_dataset_ccf()
```

2. In the main function, Feature selection is done for top 20 features for all datasets. You can change this number if you want more or less features.



```python
# Feature selection
top_features = feature_selection(X_train, y_train, 20)
X_train = X_train[top_features]
X_test = X_test[top_features]
X_validation = X_validation[top_features]
```

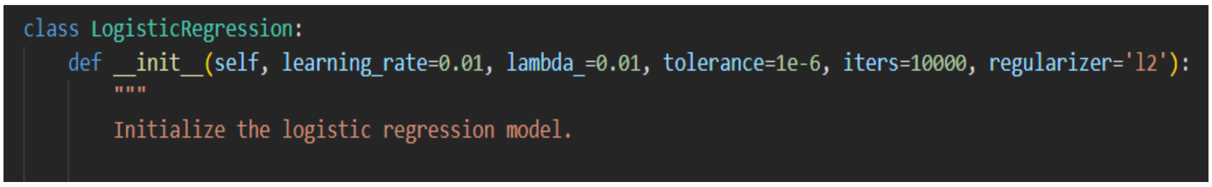
3. In performance evaluation function, for base learners and stacking L2 regularization has been used. You can use L1 if you want. (type ‘l1’)



```python
# Train 9 base learners using bagging
base_learners = train_bagging_learners(X_train, y_train, num_samples=9, regularizer='l2')

# Train the meta model using stacking
y_pred_meta = stacking_ensemble(base_learners, X_validation, y_validation, X_test, regularizer='l2')
```

4. Learning rate, lambda, iters have been used as such, you can change here if you want. Or you can change these parameters when you are calling the logistic regression from bagging or stacking.

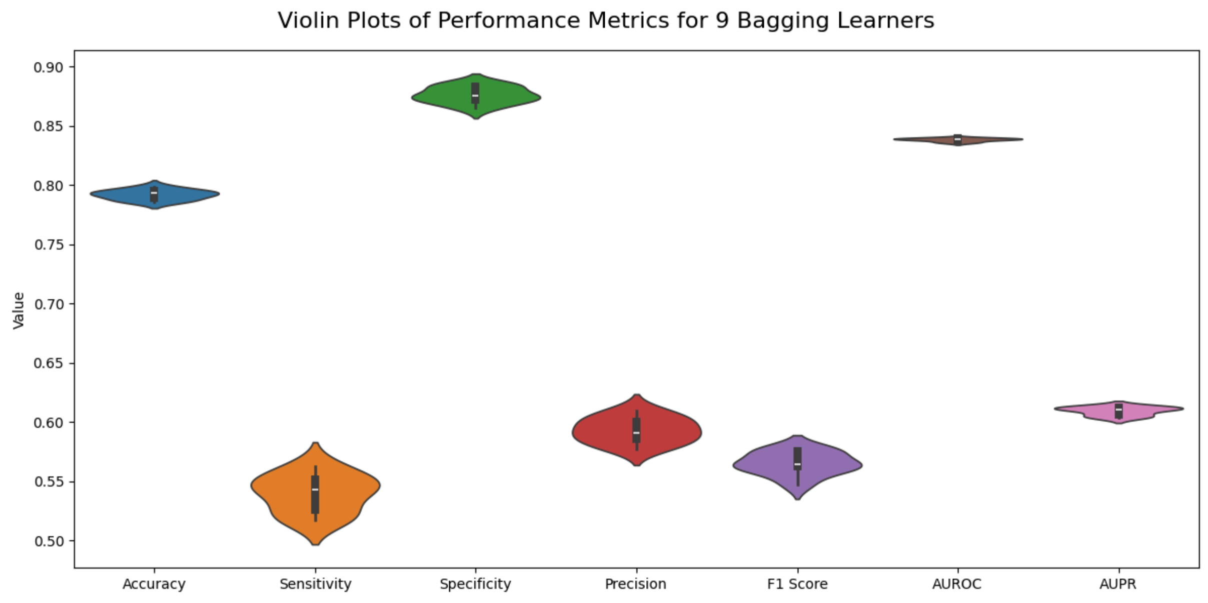


```python
class LogisticRegression:
    def __init__(self, learning_rate=0.01, lambda_=0.01, tolerance=1e-6, iters=10000, regularizer='l2'):
        """
        Initialize the logistic regression model.
```

Telco Customer Churn Dataset

<u>Hyperparameters :</u>
Learning rate = 0.01, **Lambda** = 0.01,
Tolerance = 1e-6, **Regularization** = L2, **Iterations** = 10000

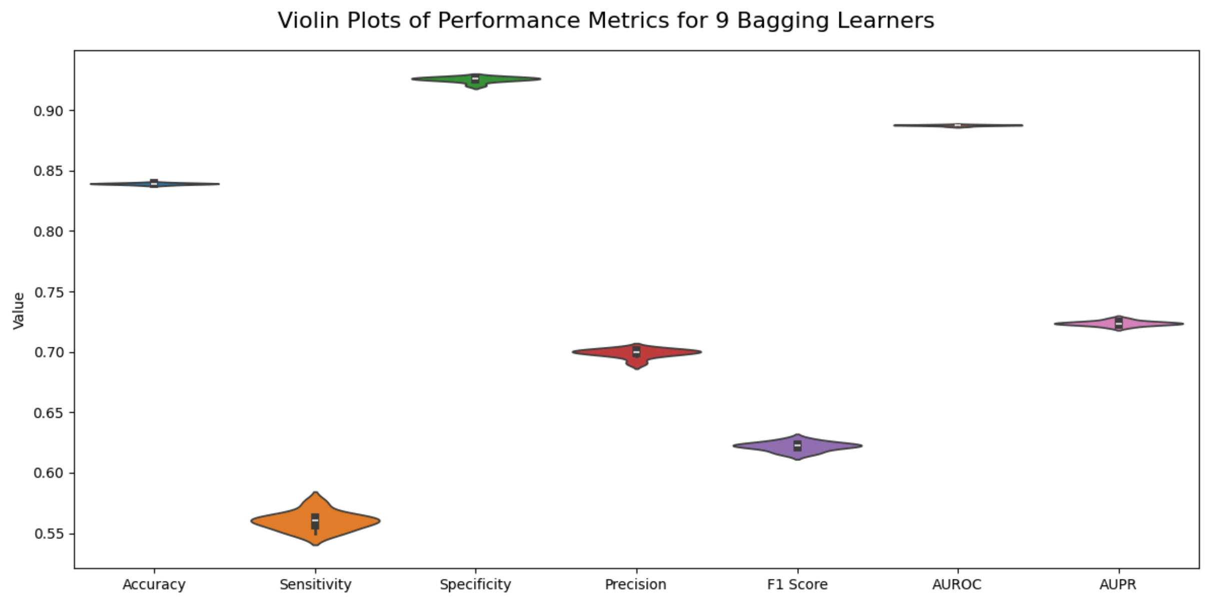
	Accuracy	Sensitivity	Specificity	Precision	F1-score	AUROC	AUPR
LR	0.7919 ± 0.0040	0.5398 ± 0.0149	0.8761 ± 0.0065	0.5931 ± 0.0101	0.5650 ± 0.0090	0.8383 ± 0.0013	0.6093 ± 0.0034
Voting ensemble	0.7957	0.5511	0.8775	0.6006	0.5748	0.7143	0.4435
Stacking ensemble	0.7815	0.5625	0.8547	0.5641	0.5633	0.7086	0.4269



Adult Salary Dataset

Hyperparameters :
Learning rate = 0.01, **Lambda** = 0.01,
Tolerance = 1e-6, **Regularization** = L2, **Iterations** = 10000

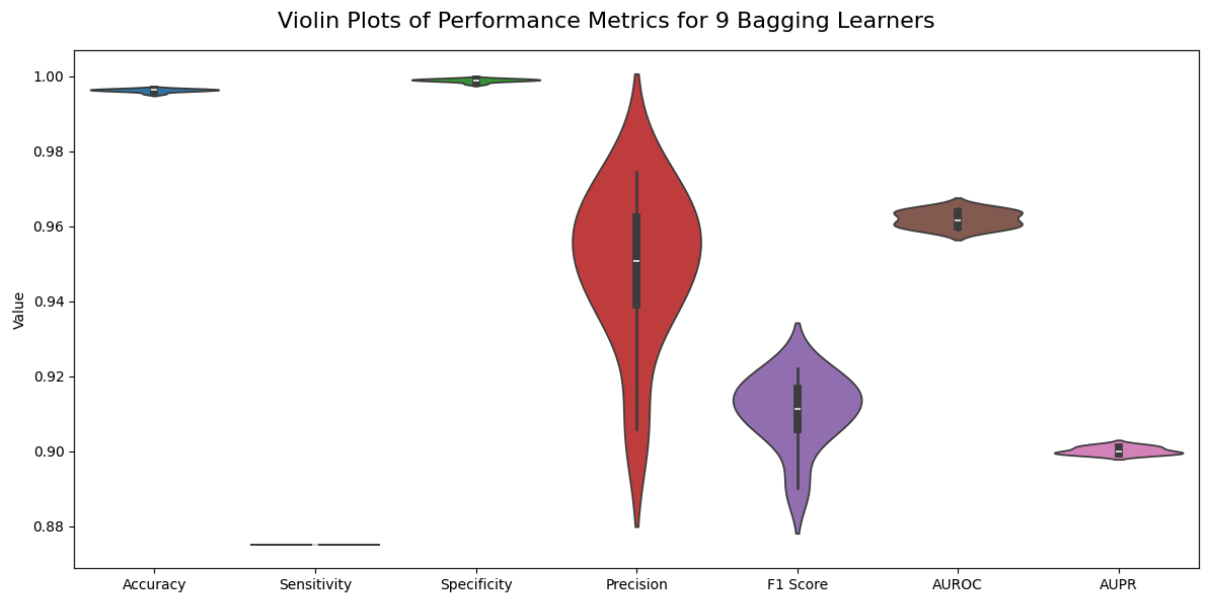
	Accuracy	Sensitivity	Specificity	Precision	F1-score	AUROC	AUPR
LR	0.8391 ± 0.0006	0.5602 ± 0.0090	0.9253 ± 0.0020	0.6990 ± 0.0033	0.6219 ± 0.0033	0.8874 ± 0.0005	0.7237 ± 0.0018
Voting ensemble	0.8396	0.5619	0.9256	0.7003	0.6235	0.7437	0.4970
Stacking ensemble	0.8394	0.5601	0.9258	0.7003	0.6224	0.7429	0.4961



Credit Card Fraud Dataset

Hyperparameters :
Learning rate = 0.01, **Lambda** = 0.01,
Tolerance = 1e-6, **Regularization** = L2, **Iterations** = 10000

	Accuracy	Sensitivity	Specificity	Precision	F1-score	AUROC	AUPR
LR	0.9963 ± 0.0004	0.8750 ± 0.0000	0.9990 ± 0.0004	0.9497 ± 0.0190	0.9107 ± 0.0089	0.9620 ± 0.0020	0.9620 ± 0.0020
Voting ensemble	0.9963	0.8750	0.9990	0.9506	0.9112	0.9370	0.8345
Stacking ensemble	0.9968	0.8750	0.9995	0.9747	0.9222	0.9373	0.8555



Observations

1. **In the first two datasets, the voting ensemble performs slightly better than the stacking ensemble**:
- **Observation**: Voting works well because the base learners are strong and diverse, capturing the dataset's patterns effectively.
- **Reason**: Majority voting efficiently aggregates predictions when the individual learners perform well on their own.
- **Comment**: The dataset likely has well-defined patterns that base learners can independently capture, making voting a simpler yet effective approach.

2. **When the iterations are reduced, the performance of the voting ensemble drops, and stacking then performs better**:
- **Observation**: With fewer iterations, base learners become weaker, impacting the voting ensemble's performance.
- **Reason**: Stacking's meta-learner can optimize how to combine weaker base models, compensating for their individual shortcomings.
- **Comment**: This suggests that the dataset requires more iterations for base learners to perform well individually, and stacking becomes advantageous when learners are undertrained.

3. **The third dataset gives better results for stacking ensemble than voting ensemble**:
- **Observation**: Stacking consistently outperforms voting in this dataset, indicating a need for a more sophisticated combination of base learners.
- **Reason**: The stacking meta-learner is able to learn the complex relationships in the data that voting cannot capture through simple aggregation.
- **Comment**: The dataset likely has more complex or non-linear patterns that require stacking’s meta-learning approach for optimal performance.

4. **If the learning rate is increased, performance metrics drop proportionately**:
- **Observation**: Higher learning rates cause performance to deteriorate for both voting and stacking ensembles.
- **Reason**: Increasing the learning rate can lead to overshooting during training, preventing the models from converging properly.
- **Comment**: This indicates that the models are sensitive to hyperparameter tuning, and higher learning rates disrupt their ability to learn from the data effectively.