Assignment
Prediction of DNA Binders

Group 25 : Nikhil Kolla MT19123
Arnab Chatterjee MT19130
Kastala Murali Krishna MT19132

Input : Amino Acid Sequence
Output : Should predict whether the given sequence is DNA binding protein or Non DNA binding protein.

Feature Extraction

An amino acid sequence is given which is a string. For Machine learning models to learn from the training set, we should convert these strings into vectors. This conversion of string to vectors is called feature extraction. Here we have used 3 different feature extraction techniques :

a. Amino Acid : It contains 20 vectors. And these are the 20 amino acid characters.
b. Dipeptide : It contains 400 vectors. Two consecutive amino acids form dipeptide.
c. Dipeptide with 780 vectors : Here we even take reverse of each dipeptide sequence. So we will get (400 + 400 - 20) = 780 vectors.
d. Tripeptide : It contains 8000 vectors. Three consecutive amino acids form tripeptides.

Normalization

We have tried two different normalization methods :

a. StandardScalar()
b. MinMaxScalar()

But we have achieved better results without using any predefined normalization method because while calculating feature extraction we divide with the length string.

Feature Reduction

Feature reduction is only used for tripeptides because it has 8000 vectors. Methods used to reduce features are :

a. Principal Component Analysis (PCA) : We have reduced the features to 500 vectors.

Models Applied

We have applied almost all the machine learning models :

a. Random Forest
b. SVM
c. Decision Trees
d. K-Nearest Neighbours
e. Naive Bayes
f. Ensemble Methods : Applied from the top three models which got high accuracy
g. CNN

Results :

From the training data, we have done 70:30 split to obtain validation data.

3 Python files were uploaded with different feature extraction methods .

Note : As we have limited submissions in kaggle, we have not uploaded all the predicted files into kaggle. So you may find blank in the kaggle score section.

a. Amino Acid Feature Extraction

Model	Accuracy of validation data	Kaggle Score
SVM	71.69	70.093
Decision Tree	71.25	69.995
KNN	67.77	-
Naive bayes	68.30	-
XGBoost	70.163	69.158
Ensemble Technique (SVM,KNN,NB)	Done on test data	70.280

CNN	65.53	-

ROC_AUC_Score is only calculated for those models which got the highest kaggle score.

Ensemble Technique	Roc_score : 69.36	70.280

b. Dipeptide Feature Extraction - 400 features

Model	Accuracy	Kaggle Score
SVM	59.23	54.205
Decision Tree	62.75	53.56
KNN	59.76	54.20
Random Forest	65.79	59.253
Naive bayes	68.80	66.355
Ensemble Technique (SVM,KNN,NB)	Done on test data	55.32
CNN	67.74	-

We have not achieved top score using dipeptide feature extraction, that is the reason we have not calculated the auc score.

c. **Dipeptide with 780 features**

Model	Accuracy	Kaggle Score
SVM	67.65	68.78
XGBoost	72.89	70.280
KNN	67.11	60.79
Random Forest	66.44	57.75
Naive bayes	68.30	67.85

Ensemble Technique (SVM,KNN,NB)	Done on test data	69.92
CNN	Not done	-

Here we XGBoost have achieved the highest score i.e 70.280 and ROC score is calculated.

XGBoost	Roc score: 68.97	70.280

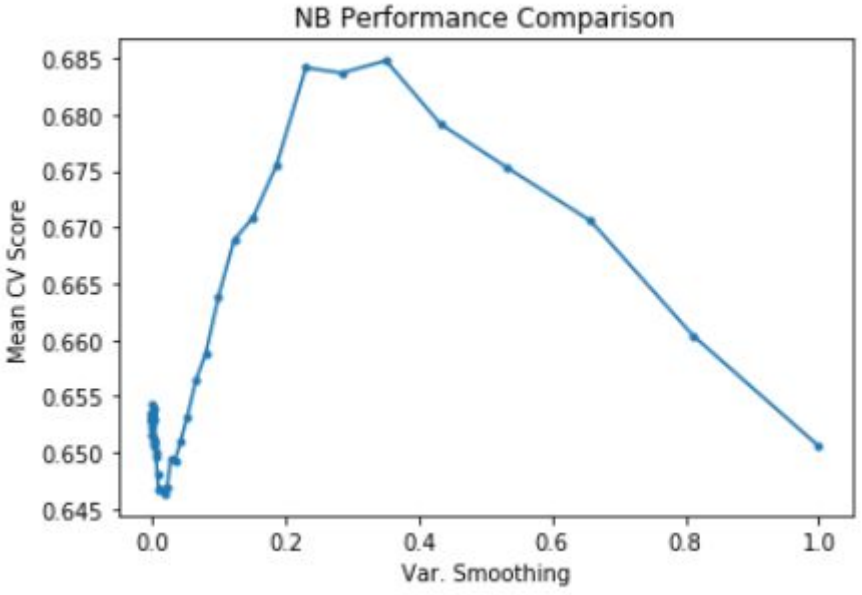
d. **Tripeptide with 8000 sequence (Features reduced to 500)**

We have not applied more models on 8000 features because it yielded very less results as compared to other feature extractions.

Model	Accuracy	Kaggle Score
KNN	67.23	55.67
Random Forest	65.72	57.75
Naive bayes	63.88	56.76

More ScreenShots from the python file :

- Naive Bayes GridSearchCV with different VarSmoothing values :



In Kaggle public scoreboard, SVM and XGBoost with amino acid feature selections and dipeptides feature selection has yielded highest accuracy. These models code snippet is given below :

- SVM with GridSearchCV and Best Parameters :

```
svm_model = Pipeline([('clf', SVC())])
param_C = [0.0001, 0.001, 0.01, 0.1, 1.0, 10.0]
param_scoring = ["accuracy"]

param = [{'clf__C': param_C,'clf__kernel': ['linear','rbf']}]
grid = GridSearchCV(estimator=svm_model,param_grid=param,refit = 'accuracy',scoring=param_scoring,verbose = 100,cv=10,n_jobs=-1)
grid = grid.fit(x_train,y_train)
```

This Yielded us the best score in Kaggle.

```
print(" The Accuracy of the best model is : ",grid.best_score_)
print(" Best Parameters : ",grid.best_params_)
```

```
The Accuracy of the best model is :  0.7240094774252993
Best Parameters :  {'clf__C': 1.0, 'clf__kernel': 'rbf'}
```

ROC Score Calculation :

```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_val,grid.predict(x_val))
```

```
0.6899227685442987
```

- XGBoost Model with Gridsearchcv (Best Model) :

```
estimator = XGBClassifier(
    objective= 'binary:logistic',
    nthread=4,
    seed=42
)
```

```
parameters = {
    'max_depth': range (4, 10, 1),
    'n_estimators': [100,200,400,600],
    'learning_rate': [0.1, 0.01]
}
```

```
grid_search = GridSearchCV(
    estimator=estimator,
    param_grid=parameters,
    scoring = 'accuracy',
    n_jobs = 10,
    cv = 10,
    verbose=100
)
```

```
grid_search.fit(x_train, y_train)
```

```
grid_search.best_estimator_
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=7,
              min_child_weight=1, missing=None, n_estimators=600, n_jobs=1,
              nthread=4, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=42,
              silent=None, subsample=1, verbosity=1)
```

```
grid_search.best_score_
```

```
0.7385393357026896
```

ROC Score :

```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_val,YBoostValidate)
```

```
0.6936407766990291
```

- In ensemble methods, we have used a voting classifier on the test data, which again yielded the same accuracy that is 70.280

As in the kaggle challenge we can submit fill three top scores to calculate the private score, We have got top score from AminoAcid feature selection and Dipeptide feature selection. So Two .py files are uploaded.
