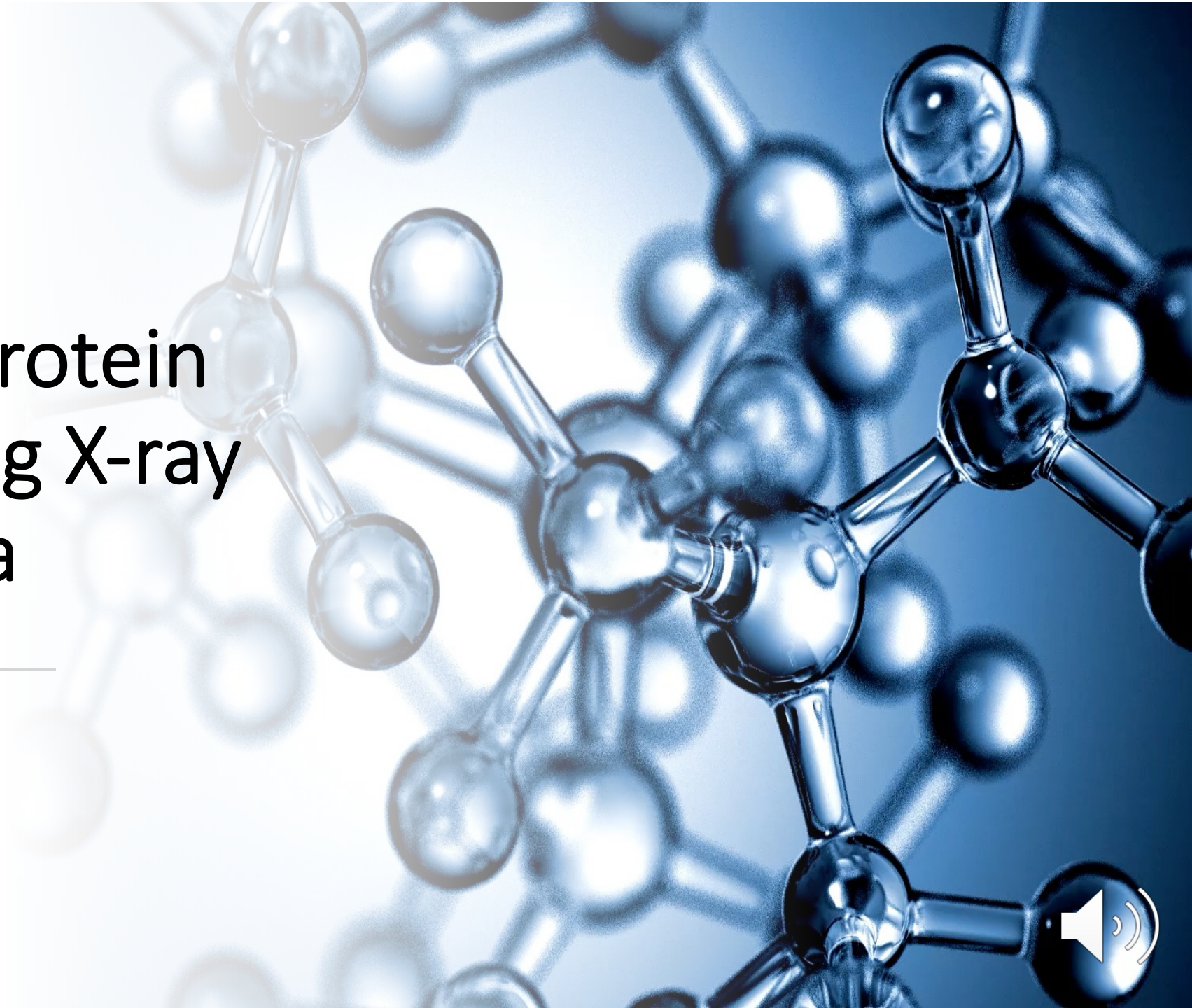
UMBC

Predicting the resolution of protein structures using X-ray diffraction data

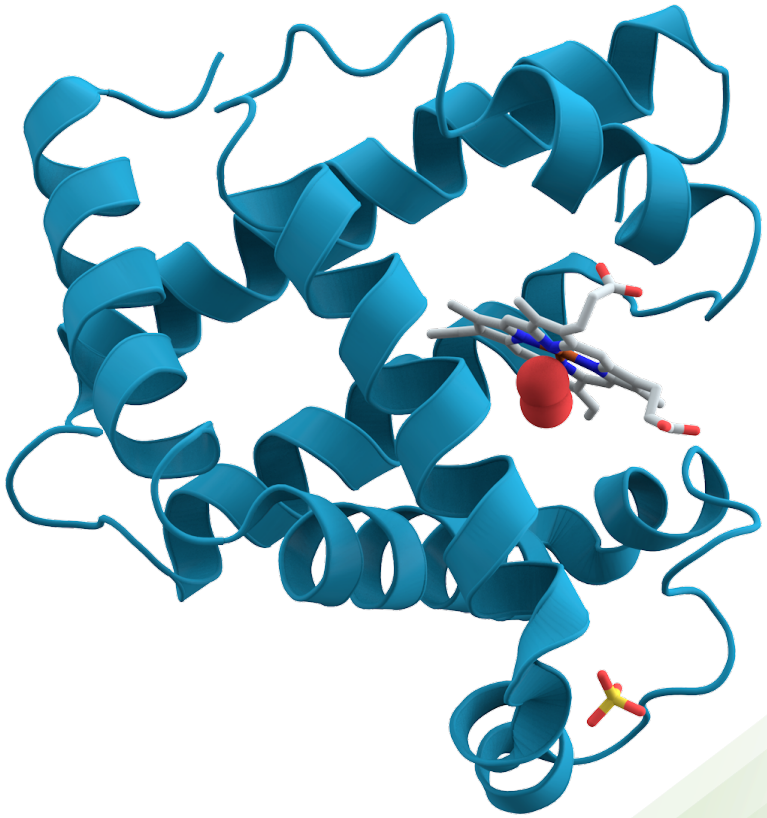


Data 602 Final Project

Shreya Patil
HG53212

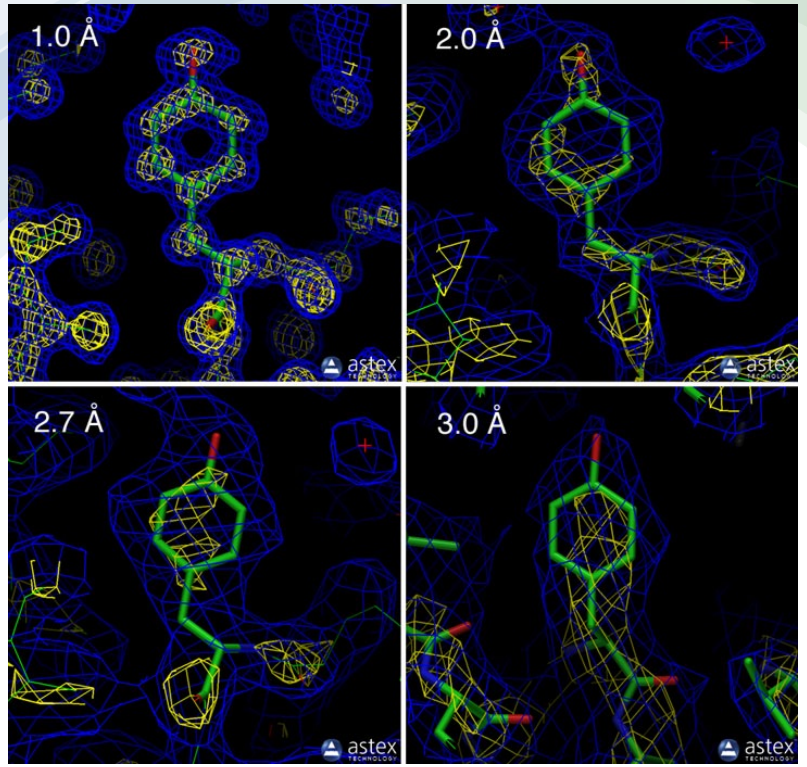
Introduction

- Protein study from atomic structures.
- Applications in protein engineering and drug discovery.
- X- ray crystallography, NMR and Cryo-EM are different methods.
- Factors affecting quality of resolution
  - crystal dimensions
  - crystal properties
  - X-ray data collection details



Problem Statement

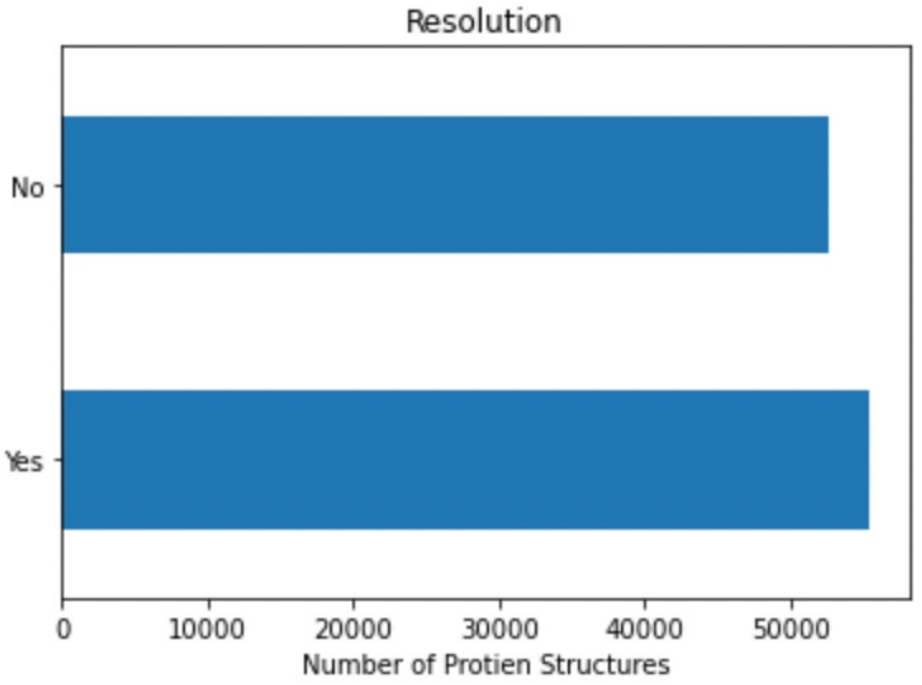
- Predicting the resolution of protein structures using X-ray diffraction data
- Resolution is the measure of details observed in a diffraction pattern.
- For this project we will be converting these resolution values into two groups as good(Yes) or bad(No) with 2 Å as threshold resolution.



Understanding the X-ray Diffraction Data of Protein Structures to Predict Resolution

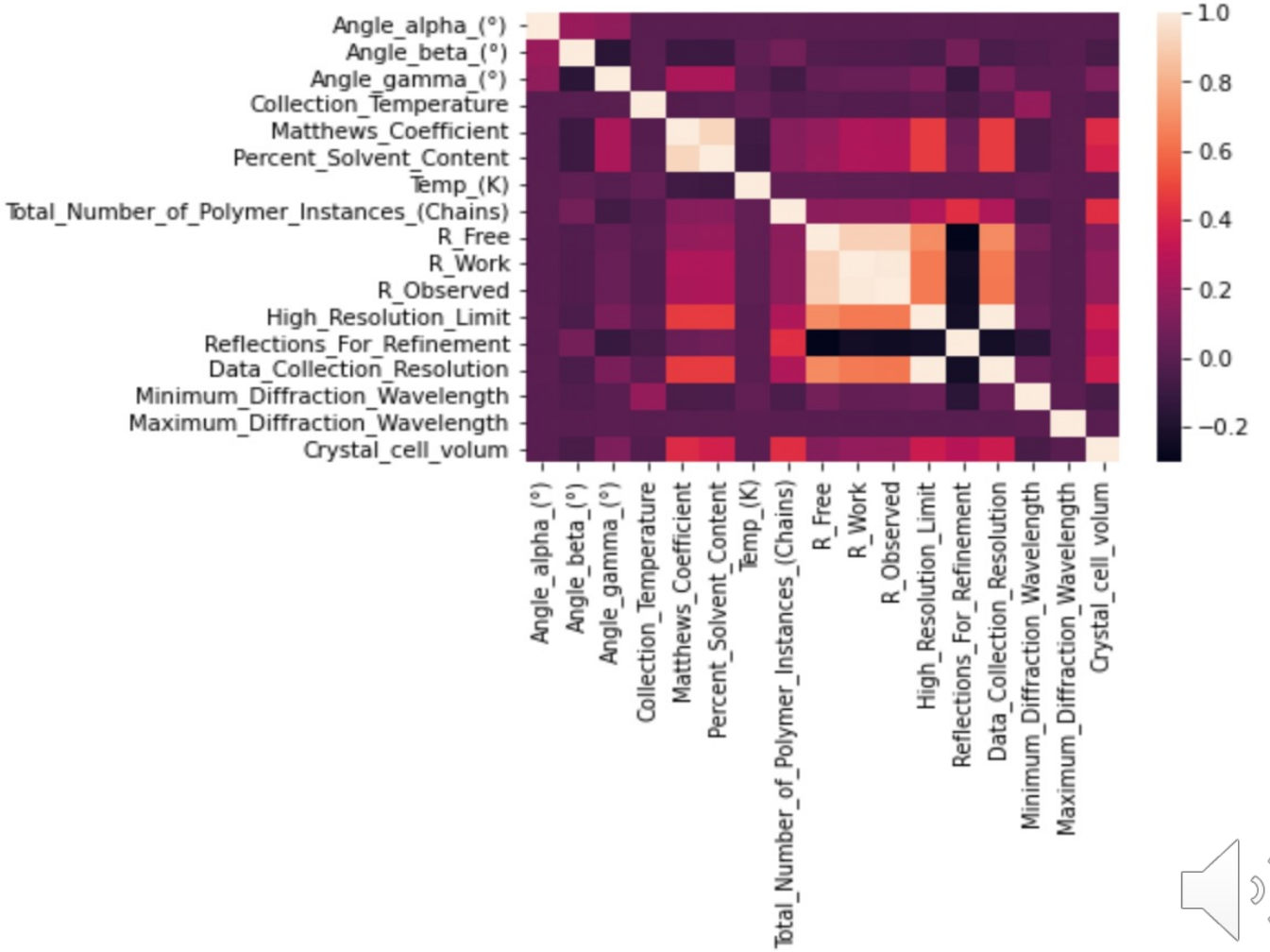
Exploratory Data Analysis

- We have dropped the unnecessary features from the dataset along with missing values observations.
- Feature-like lengths of crystal cells are converted into volume
- Target variable is converted into the required format.
- Converted the datatype of features to the appropriate values.
- Converted the categorical feature data to uppercase.
- Analyzed the correlation between numeric variables.
- Dataset Size after EDA (108064, 15)



Features causing multicollinearity

	feature	VIF
0	Angle_alpha_(°)	764.066008
1	Angle_beta_(°)	135.782582
2	Angle_gamma_(°)	76.527439
3	Collection_Temperature	9.631765
4	Matthews_Coefficient	118.721456
5	Percent_Solvent_Content	198.630808
6	Temp_(K)	701.450550
7	Total_Number_of_Polymer_Instances_(Chains)	2.479874
8	R_Free	299.881433
9	R_Work	1341.420877
10	R_Observed	1257.008216
11	High_Resolution_Limit	1993.159324
12	Reflections_For_Refinement	3.138797
13	Data_Collection_Resolution	1952.610806
14	Minimum_Diffraction_Wavelength	28.901456
15	Maximum_Diffraction_Wavelength	1.000845
16	Crystal_cell_volum	1.989524



Dataset

PDB Dataset (https://bit.ly/3K0LUJq)

```
Data columns (total 15 columns):
 #   Column                                Non-Null Count   Dtype
---  ------                                --------------   -----
 0   Angle_beta_(°)                        108064 non-null  float64
 1   Angle_gamma_(°)                       108064 non-null  float64
 2   Collection_Temperature                108064 non-null  float64
 3   Detector                              108064 non-null  object
 4   Diffraction_Source_General_Class      108064 non-null  object
 5   Matthews_Coefficient                  108064 non-null  float64
 6   Percent_Solvent_Content               108064 non-null  float64
 7   Total_Number_of_Polymer_Instances_(Chains)  108064 non-null  int64
 8   Resolution_(Å)                        108064 non-null  object
 9   R_Free                                108064 non-null  float64
 10  Reflections_For_Refinement            108064 non-null  float64
 11  Structure_Determination_Method        108064 non-null  object
 12  Minimum_Diffraction_Wavelength        108064 non-null  float64
 13  Maximum_Diffraction_Wavelength        108064 non-null  float64
 14  Crystal_cell_volum                    108064 non-null  float64
```

Modeling and Results

- **Logistic Regression with Regularization**

```
params = {'lg__C': [0.01, 0.1, 1, 10], 'lg__penalty':['l2', 'none']}
```

```
('lg', LogisticRegression(C=10))])
```

```
lg_l2_gscv.best_score_
```

```
0.8395275336390429
```

- **Secondary Hyperparameter Search**

```
params = {'lg__C': [5, 8, 10, 12,15], 'lg__penalty':['l2', 'none']}
```

```
('lg', LogisticRegression(C=10, solver='newton-cg'))])
```

```
lg_l2_gscv.best_score_
```

```
0.8397473151967885
```

	precision	recall	f1-score	support
0	0.84	0.83	0.83	10454
1	0.84	0.85	0.85	11159
accuracy			0.84	21613
macro avg	0.84	0.84	0.84	21613
weighted avg	0.84	0.84	0.84	61

Modeling and Results

- **Logistic Regression with PCA**

```
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
from sklearn.decomposition import PCA

n_comp_list=[5,6,7,8,9,10]

for i in n_comp_list:
    num_pipeline = Pipeline([('scale', StandardScaler()), ('pca', PCA(n_components=i)) ])
    car_pipeline = Pipeline([('create_dummies_cats', OneHotEncoder(handle_unknown='ignore', drop='first'))])


    processing_pipeline = ColumnTransformer(transformers=[('proc_numeric', num_pipeline, numerical_vars),
                                                           ('create_dummies', car_pipeline, categorical_vars)])


    modeling_pipeline = Pipeline([('data_processing', processing_pipeline),
                                  ('logreg', LogisticRegression(penalty='none'))])

    pca_model=modeling_pipeline.fit(X_train, y_train)
    pca_y_predicted = pca_model.predict(X_test)


    print('Accuracy for %d n_components :'%(i),accuracy_score(y_test, pca_y_predicted))
```

```
Accuracy for 5 n_components : 0.8138157590339148
Accuracy for 6 n_components : 0.8167769398047472
Accuracy for 7 n_components : 0.8163142553093046
Accuracy for 8 n_components : 0.8201082681719336
Accuracy for 9 n_components : 0.8286679313376208
Accuracy for 10 n_components : 0.8397723592282422
```

Modeling and Results

- **Decision Tree Classifier**

```
param_grid = [
  {'dt__max_depth': [2, 5, 10, 15, 20],
   'dt__min_samples_split':[0.01, 0.05, 0.10]}]
```

```
('dt',
 DecisionTreeClassifier(max_depth=10, min_samples_split=0.01))])
```

	precision	recall	f1-score	support
0	0.85	0.83	0.84	10454
1	0.84	0.86	0.85	11159
accuracy			0.84	21613
macro avg	0.84	0.84	0.84	21613
weighted avg	0.84	0.84	0.84	21613

Modeling and Results

- **Secondary Search on Decision Tree Classifier**

```
params = {'dt__max_depth': [8,9,10, 11, 12, 13],
          'dt__min_samples_split':[0.005,0.007,0.01, 0.02, 0.03]
         }
```

```
('dt',
 DecisionTreeClassifier(max_depth=10,
                        min_samples_split=0.005))])
```

	precision	recall	f1-score	support
0	0.85	0.85	0.85	10454
1	0.86	0.86	0.86	11159
accuracy			0.85	21613
macro avg	0.85	0.85	0.85	21613
weighted avg	0.85	0.85	0.85	21613

Modeling and Results

Test Data Performance

```
from sklearn.metrics import classification_report

dt_pred_evt = dt.predict(X_test)

print(classification_report(y_test, dt_pred_evt))
```

	precision	recall	f1-score	support
0	0.85	0.85	0.85	10454
1	0.86	0.86	0.86	11159
accuracy			0.85	21613
macro avg	0.85	0.85	0.85	21613
weighted avg	0.85	0.85	0.85	21613

Training Data Performance

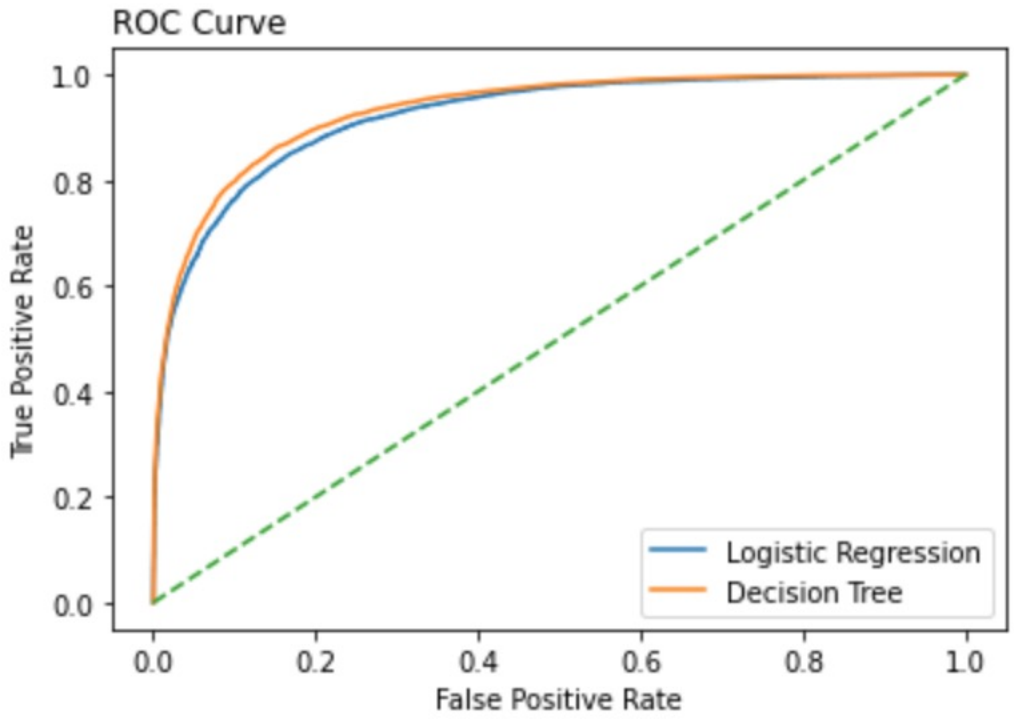
```
from sklearn.metrics import classification_report

dt_pred_evt = dt.predict(X_train)

print(classification_report(y_train, dt_pred_evt))
```

	precision	recall	f1-score	support
0	0.85	0.86	0.86	42174
1	0.87	0.86	0.86	44277
accuracy			0.86	86451
macro avg	0.86	0.86	0.86	86451
weighted avg	0.86	0.86	0.86	86451

Modeling and Results




```
Time taken by Logistic Regression Model: 820.7268430000004
Time taken by Decision Tree Model: 121.97776299999987
```

Conclusion and Future Scope

- The Decision Tree Classifier is best the performing model.
- Accuracy of the model is 85% which proves the relation between features and target value.
- Using additional features related to X-ray diffraction data.
- Using Ensemble Methods to improve the results.
- Use of Multiclass Classification model.

Thank You !