Data Analysis Report
for Molecular Biology Splice Junction gene sequences

Speaker's Name, Abhinav Dogra
16-03-2023

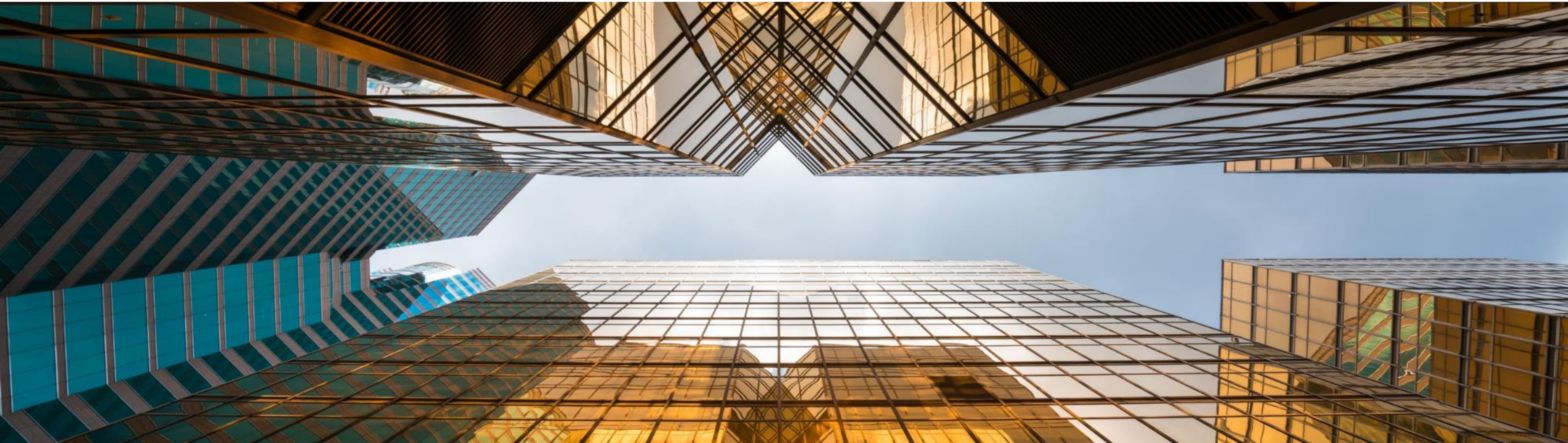
Molecular_Biology_Splice_Junction_Gene_Sequences review Analysis

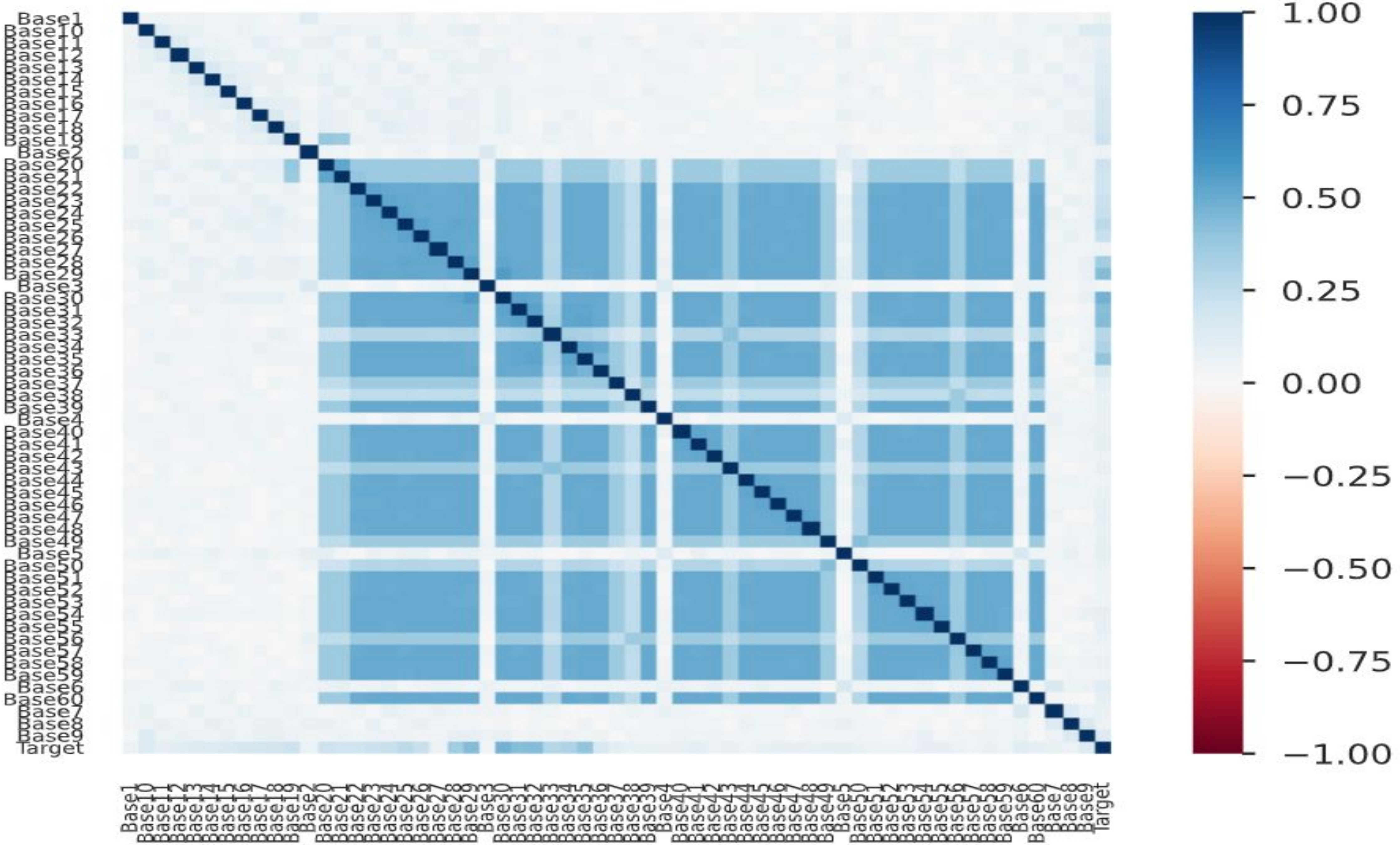
- Dataset Information
  - RangeIndex: 3190 entries
  - Columns: 61 { 0-59 Feature, 1 Target)
  - Null values: 0
  - Duplicated Rows: 10

- Unique Target Category
  - ['EI', 'IE', 'N']

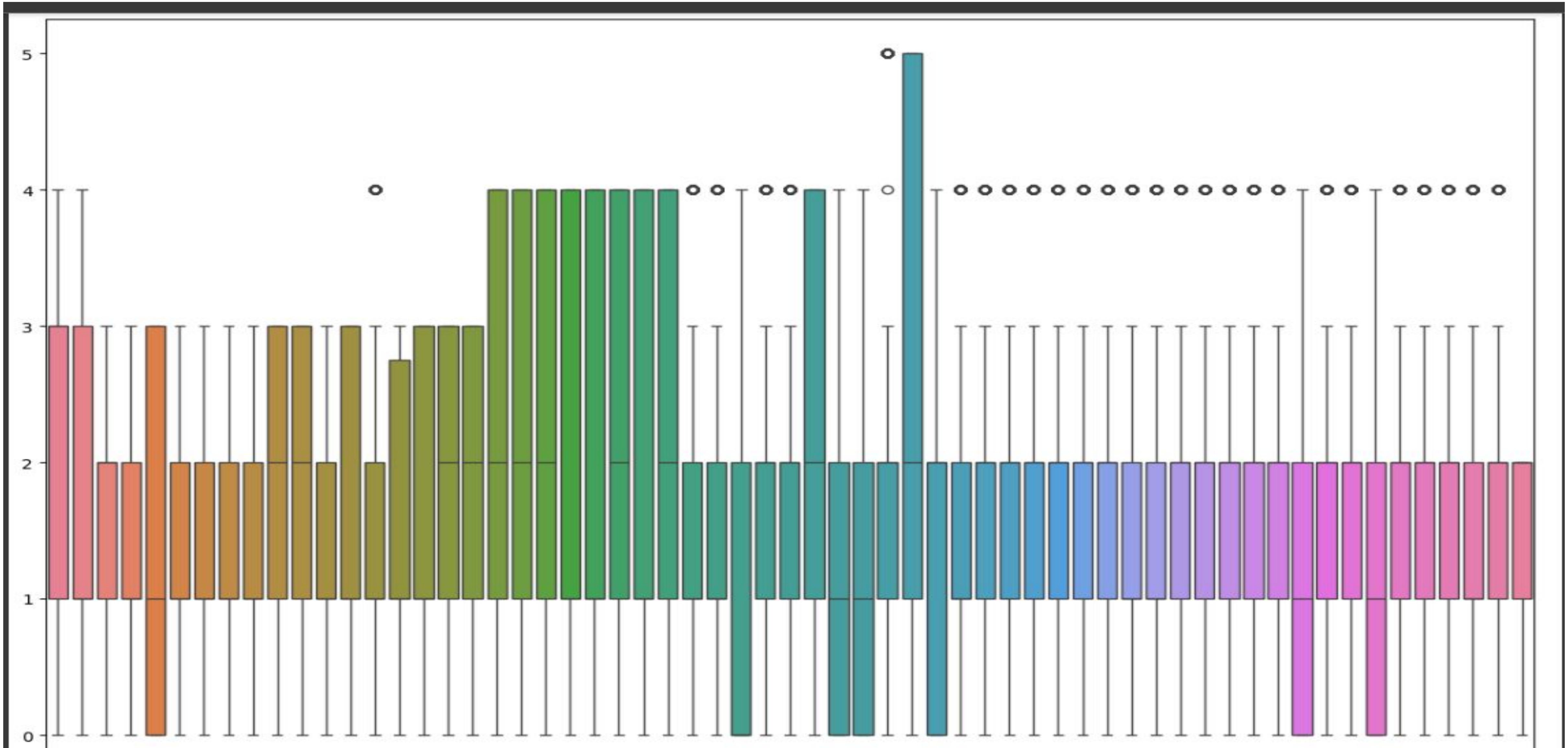
```
N       1655
IE       768
EI       767
Name: Target, dtype: int64
```

Features per Target




Base1
Base10
Base11
Base12
Base13
Base14
Base15
Base16
Base17
Base18
Base19
Base2
Base20
Base21
Base22
Base23
Base24
Base25
Base26
Base27
Base28
Base29
Base3
Base30
Base31
Base32
Base33
Base34
Base35
Base36
Base37
Base38
Base39
Base4
Base40
Base41
Base42
Base43
Base44
Base45
Base46
Base47
Base48
Base49
Base5
Base50
Base51
Base52
Base53
Base54
Base55
Base56
Base57
Base58
Base59
Base6
Base60
Base7
Base8
Base9
Target
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

Box Plot: Dimensionality of data

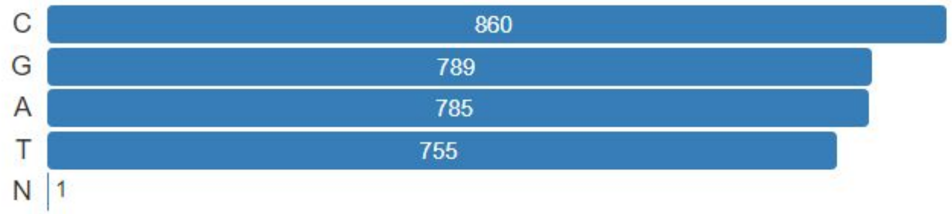


Base59

Categorical

HIGH CORRELATION

Distinct	5
Distinct (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	25.0 KiB



More details

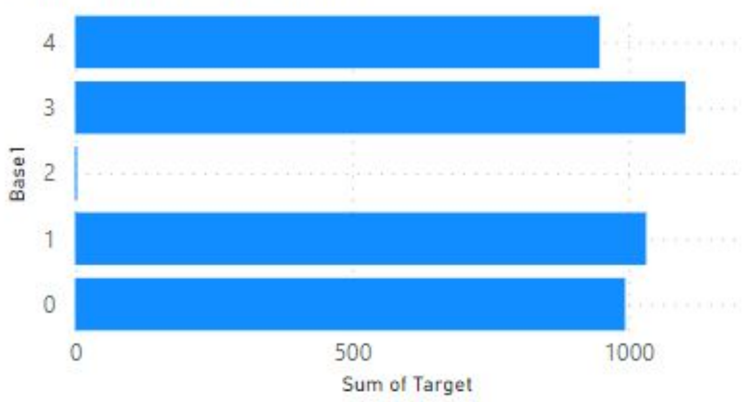
Overview Categories Words Characters

Common Values

Value	Count	Frequency (%)
C	860	27.0%
G	789	24.7%
A	785	24.6%
T	755	23.7%
N	1	< 0.1%

Length

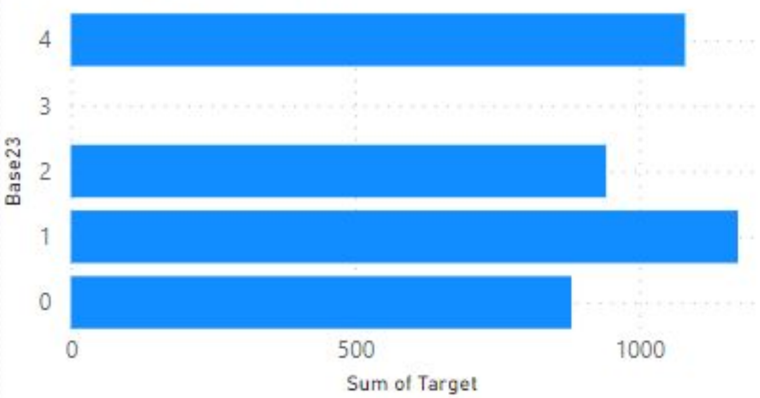
Sum of Target by Base1



Sum of Target by Base20

Sum of Target by Base21

Sum of Target by Base23

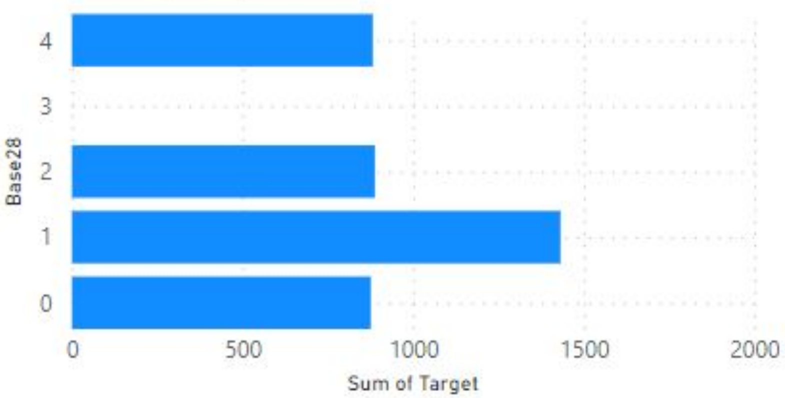


Sum of Target by Base24

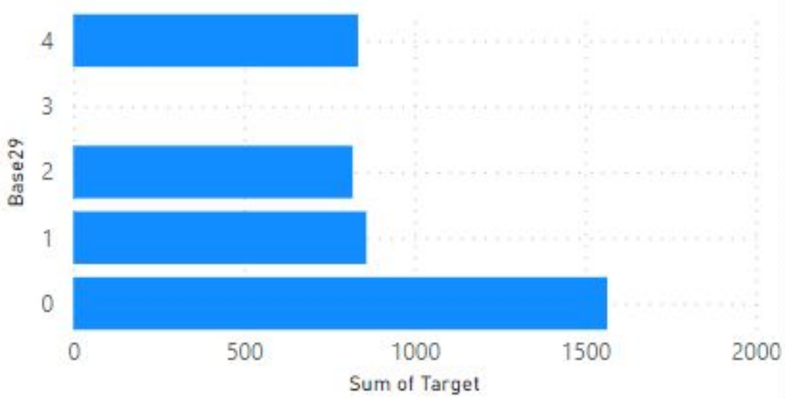
Sum of Target by Base26

Sum of Target by Base27

Sum of Target by Base28



Sum of Target by Base29



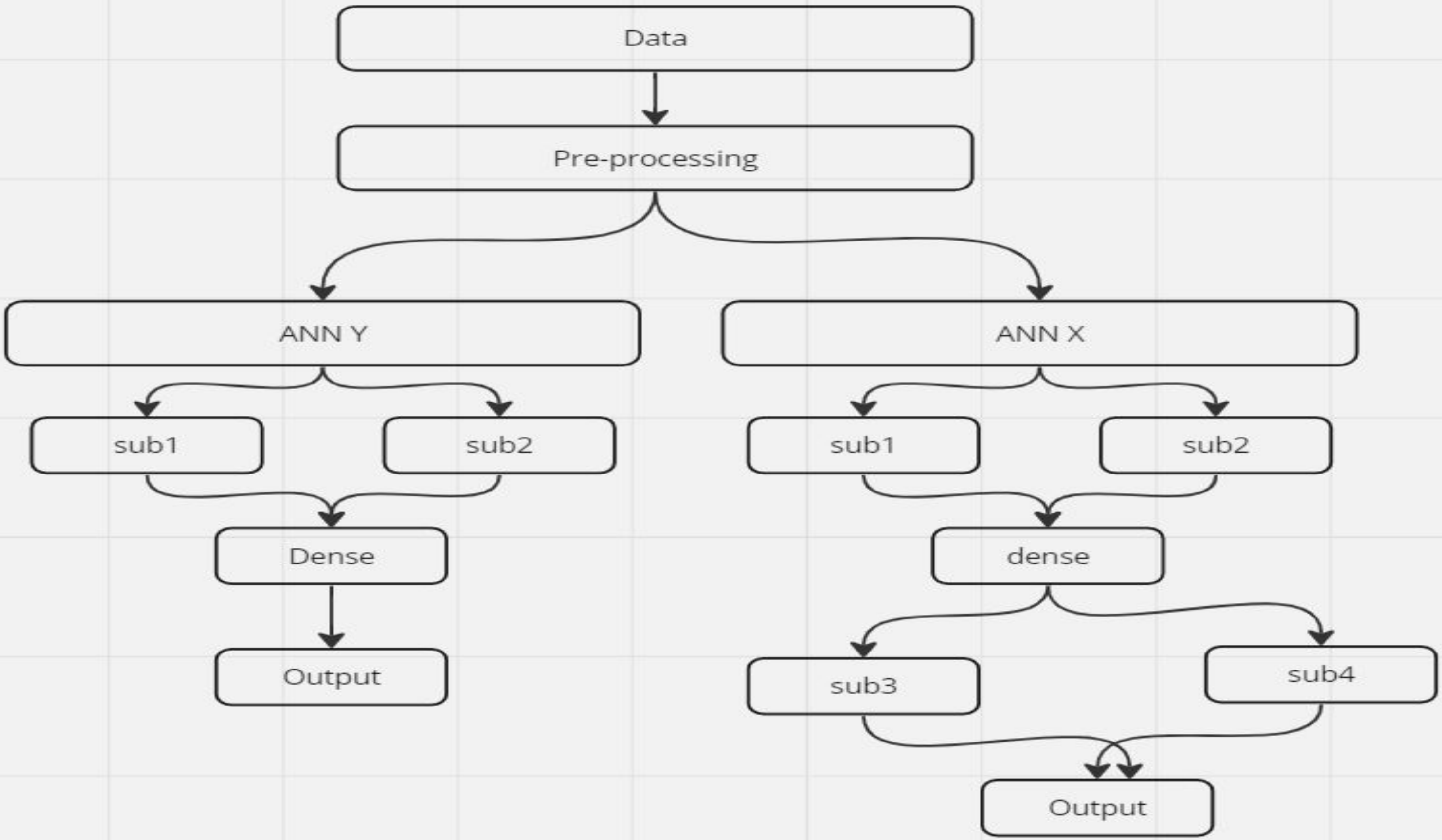
Preprocessing

Preprocessing the data

```
[ ] def preproc_data(df, train_sample: float, pca_dim=31):

        # Label encode
        categorical_cols = df.select_dtypes(include=['object']).columns

        # If there are categorical columns, encode them
        if len(categorical_cols) > 0:
            label_encoder = LabelEncoder()
            for col in categorical_cols:
                df[col] = label_encoder.fit_transform(df[col])

        # Train test split
        x_train, x_test, y_train, y_test = train_test_split(df.iloc[:,:-1],
                                                            df['Target'],
                                                            test_size=1-train_sample,
                                                            random_state=0)
        # Standard scaling
        ss = StandardScaler().fit(x_train)

        x_train = ss.transform(x_train)
        x_test = ss.transform(x_test)

        # PCA
        pca = PCA(n_components=0.99).fit(x_train)

        x_train = pca.transform(x_train)
        x_test = pca.transform(x_test)

        # Normalization
        norm = Normalizer().fit(x_train)

        x_train = norm.transform(x_train)
        x_test = norm.transform(x_test)

        # Reshaping
        y_train = y_train.values.reshape(-1,1)
        y_test = y_test.values.reshape(-1,1)

        return x_train, x_test, y_train, y_test
```

data divided in 75/25 split

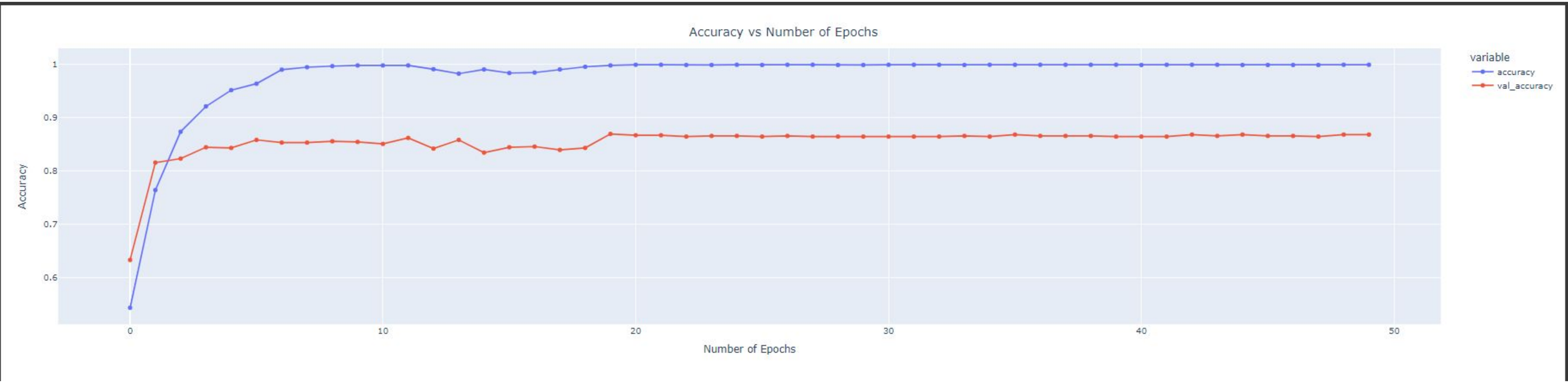
```
[ ] x_train, x_test, y_train, y_test = preproc_data(df, train_sample=0.75)# data - clean
```

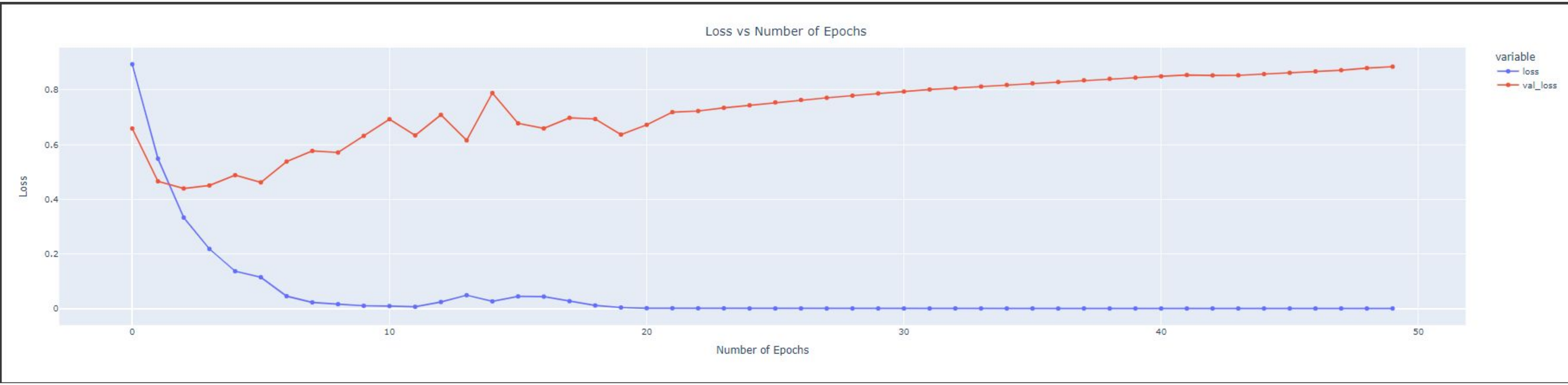

Data
Pre-processing
ANN Y
ANN X
sub1
sub2
sub1
sub2
Dense
dense
Output
sub3
sub4
Output

ANN Model Pre

```python
# Assuming you have x_train, y_train, x_test, y_test
unique_labels = len(np.unique(y_train))



y_train1_encoded = to_categorical(y_train, num_classes=unique_labels)


y_test_encoded = to_categorical(y_test, num_classes=unique_labels)




# Define the first sub-model
def sub_model1(inputs):
    x = Dense(128, activation='relu')(inputs)
    x = Dense(512, activation='relu')(x)
    x = Dense(1024, activation='relu')(x)
    x = Dense(256, activation='relu')(x)
    x = Dense(64, activation='relu')(x)
    return x

# Define the second sub-model
def sub_model2(inputs):
    x = Dense(256, activation='relu')(inputs)
    x = Dense(1024, activation='relu')(x)
    x = Dense(2048, activation='relu')(x)
    x = Dense(512, activation='relu')(x)
    x = Dense(128, activation='relu')(x)
    x = Dense(64, activation='relu')(x)
    return x


input_layer = Input(shape=(x_train.shape[1],))

sub_model1_output = sub_model1(input_layer)

sub_model2_output = sub_model2(input_layer)

concatenated_output = Concatenate()([sub_model1_output, sub_model2_output])
x = Dense(32, activation='relu')(concatenated_output)
output_layer = Dense(unique_labels, activation='softmax')(x)

model = Model(inputs=input_layer, outputs=output_layer)

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
# model.fit(X_train2, y_train1, epochs=10, batch_size=32, validation_data=(x_test, y_test))
```

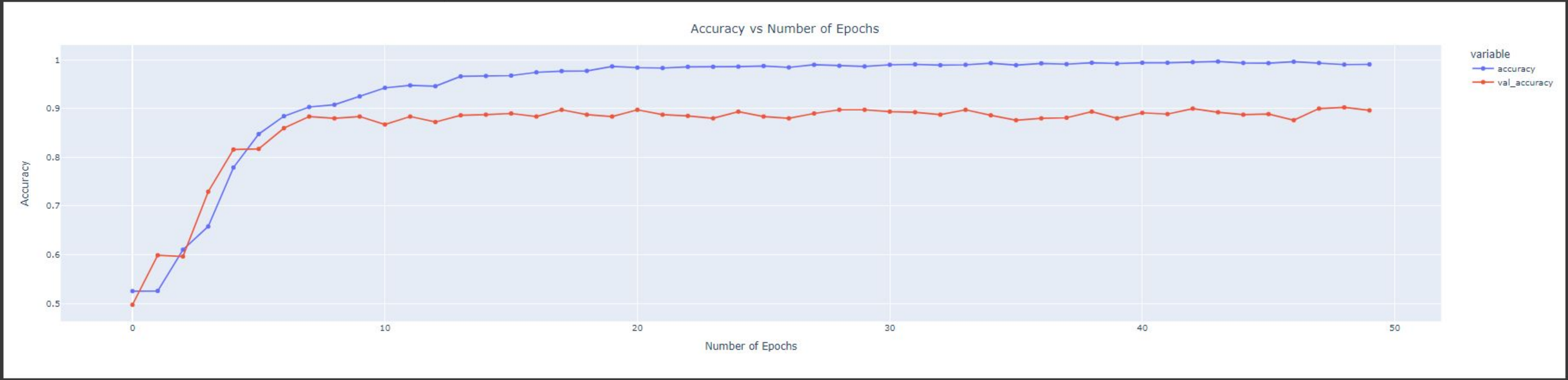
Accuracy and loss

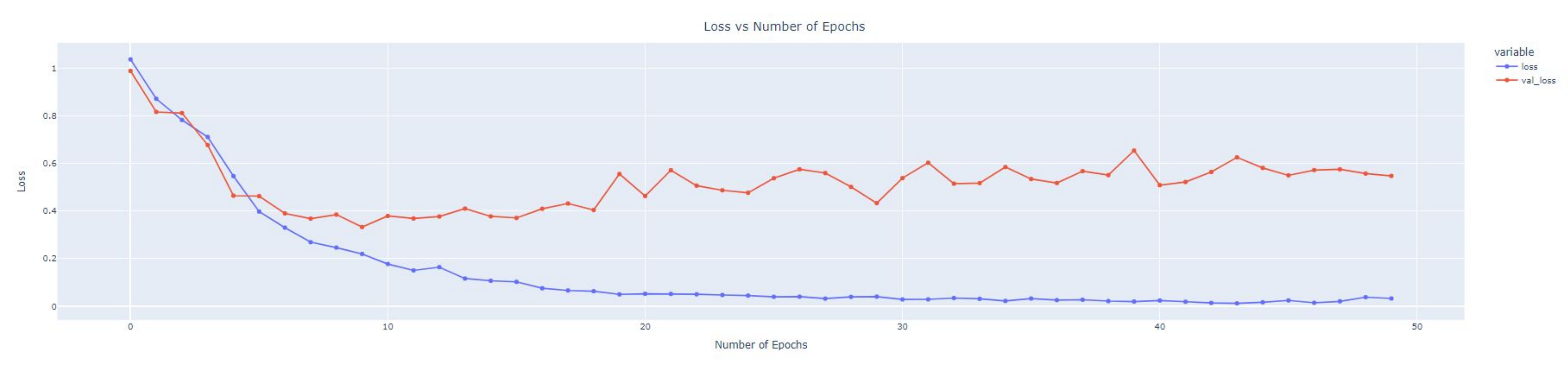




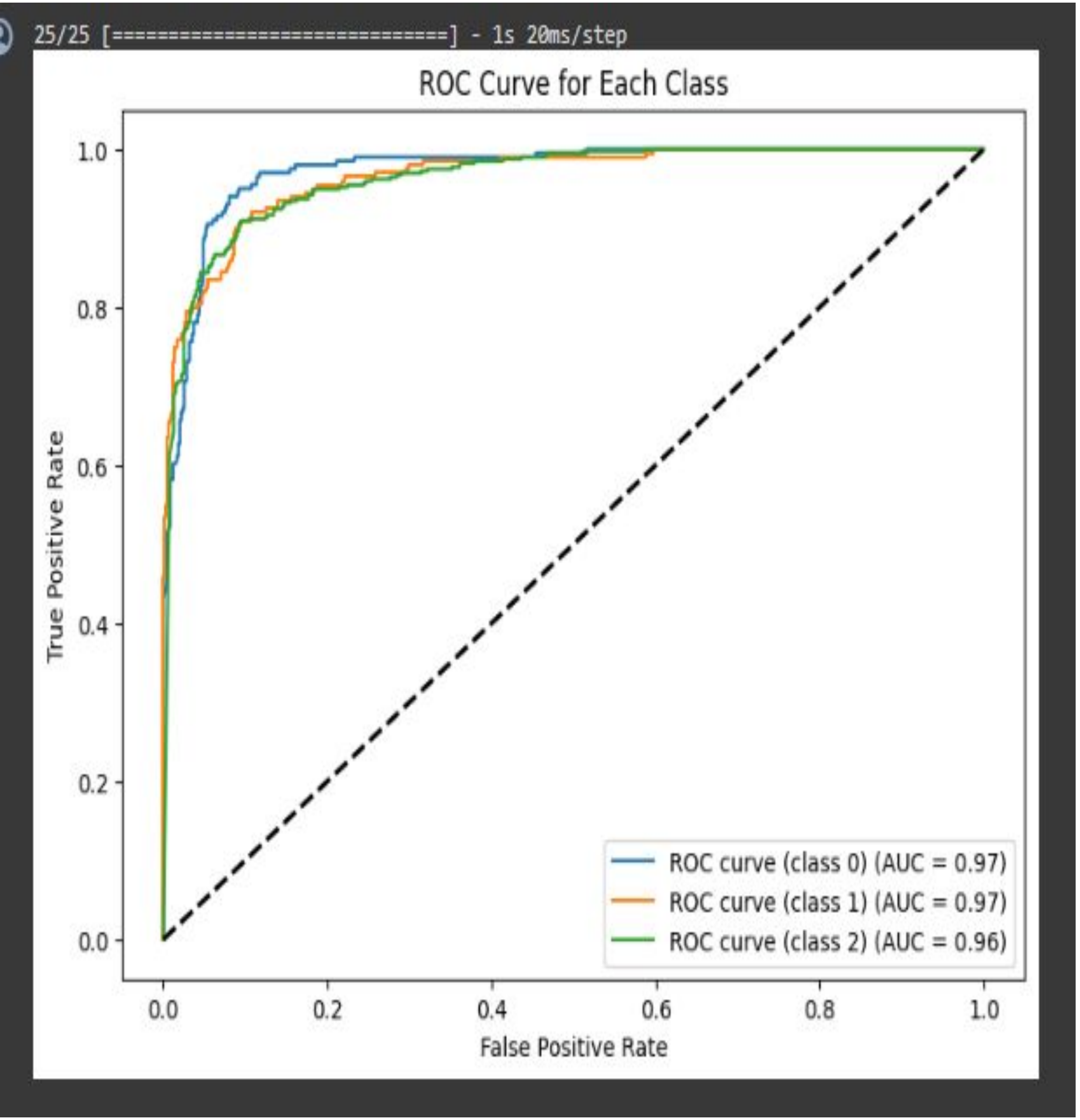
ANN model 2

```python
def sub_model1(inputs):
    x = Dense(256, activation='relu')(inputs)
    x = Dropout(0.3)(x)
    x = Dense(512, activation='relu')(x)
    x = Dropout(0.3)(x)
    x = Dense(128, activation='relu')(x)
    x = Dropout(0.3)(x)
    return x

def sub_model2(inputs):
    x = Dense(256, activation='relu')(inputs)
    x = Dropout(0.3)(x)
    x = Dense(1024, activation='relu')(x)
    x = Dropout(0.3)(x)
    x = Dense(2048, activation='relu')(x)
    x = Dropout(0.2)(x)
    x = Dense(512, activation='relu')(x)
    x = Dropout(0.2)(x)
    x = Dense(128, activation='relu')(x)
    x = Dropout(0.2)(x)
    return x

def sub_model3(inputs):
    x = Dense(64, activation='relu')(inputs)
    x = Dense(512, activation='relu')(x)
```

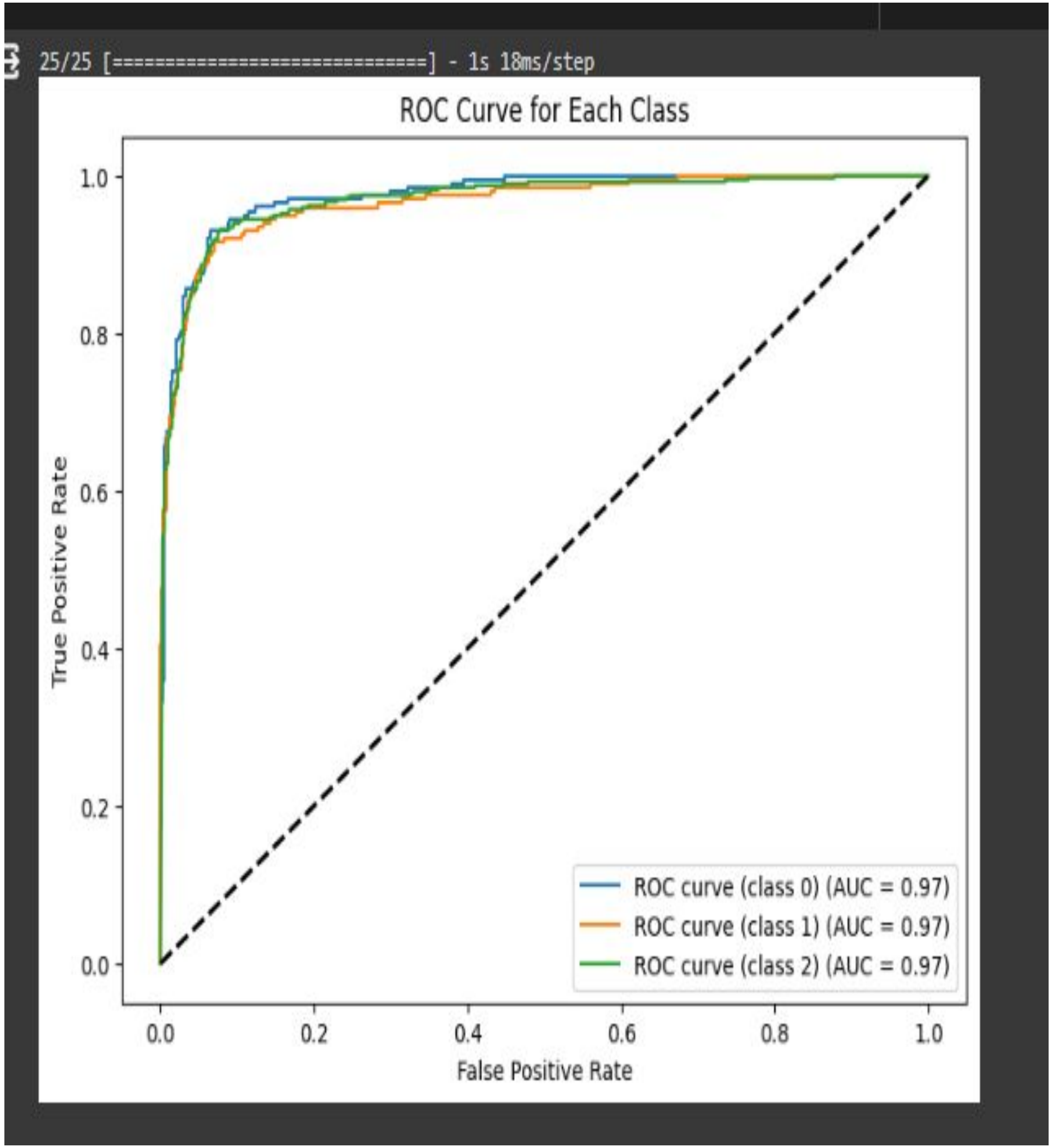

Accuracy vs Number of Epochs
Accuracy
Number of Epochs
variable
accuracy
val_accuracy
1
0.9
0.8
0.7
0.6
0.5
0
10
20
30
40
50


Loss vs Number of Epochs
Loss
Number of Epochs
variable
loss
val_loss
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40
50

model1



model2

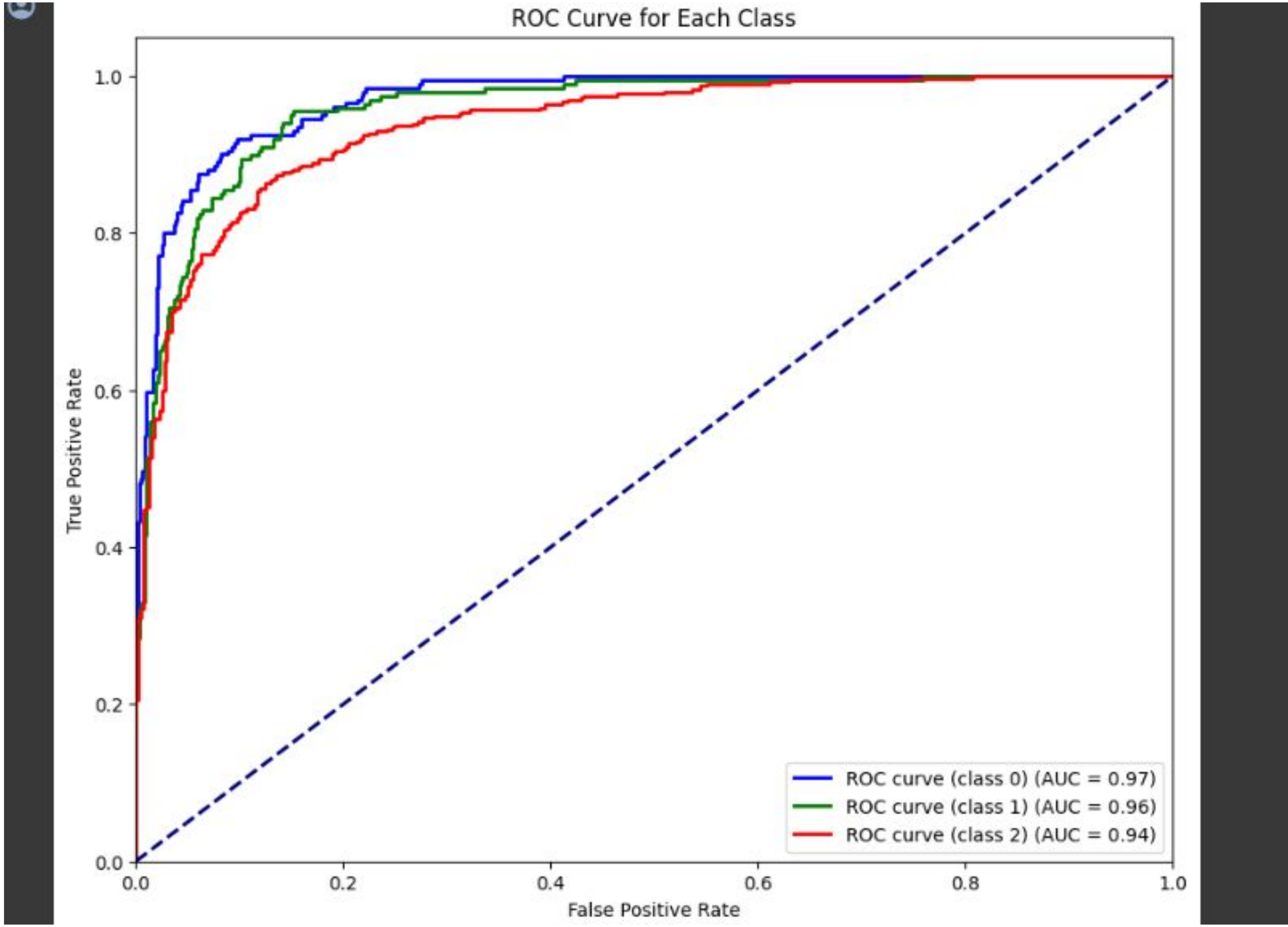


ML + Voting Classifier + Cross validation

```
svm_classifier = SVC(C=10, kernel='rbf', gamma='auto', probability=True)
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
lr_classifier = LogisticRegression(random_state=42)


voting_classifier = VotingClassifier(estimators=[
    ('svm', svm_classifier),
    ('rf', rf_classifier),
    ('lr', lr_classifier)
], voting='soft')


k_fold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

scores = cross_val_score(voting_classifier, x_train, y_train, cv=k_fold, scoring='accuracy')

print("Average Accuracy:", scores.mean())

voting_classifier.fit(x_train, y_train)

y_pred_ensemble = voting_classifier.predict(x_test)

accuracy_ensemble = accuracy_score(y_test, y_pred_ensemble)
print("Ensemble Accuracy on Test Dataset:", accuracy_ensemble)
```

ML Model processed

Creating the complex voting classifier with input model logistic regression, SVM, Random Forest with cross validation,
Average Accuracy:
0.818551550038871
Ensemble Accuracy on Test Dataset:
0.8521303258145363



H2O Model

H2o with cross validation, MSE 0.35 for cv-4

		mean	sd	cv_1_valid	cv_2_valid	cv_3_valid	cv_4_valid	cv_5_valid
0	aic	NaN	0.000000	NaN	NaN	NaN	NaN	NaN
1	loglikelihood	NaN	0.000000	NaN	NaN	NaN	NaN	NaN
2	mae	0.510577	0.013617	0.502544	0.523485	0.510404	0.492423	0.524029
3	mean_residual_deviance	0.384992	0.022916	0.374499	0.393836	0.392496	0.351744	0.412386
4	mse	0.384992	0.022916	0.374499	0.393836	0.392496	0.351744	0.412386
5	r2	0.432125	0.031040	0.418246	0.453587	0.413210	0.475065	0.400519
6	residual_deviance	0.384992	0.022916	0.374499	0.393836	0.392496	0.351744	0.412386
7	rmse	0.620255	0.018574	0.611963	0.627563	0.626495	0.593080	0.642173
8	rmsle	0.345565	0.014374	0.330114	0.357405	0.353712	0.329680	0.356914

H2O session _sid_9064 closed

conclusion

As After going around with dataset we concluded that the there 34 high correlation set which effect the model highly like 23,40, 41 etc.

The variation in data is high causing the model prediction loss to high after some time. So, require more data for better model.

The data predictability for target after optimization goes to, accuracy: 0.9904 - val_accuracy: 0.8960

After the pre process we have optimize both time and accuracy of model for ML ~78 - > ~85%

After the pre process we have loss reduce from ~0.8 -> ~0.5

and predicting efficiency of ~ 97% Achieved.

Thank you.

Contact information: 9205802993

Abhinav Dogra
Team : @Raven