DATASET :

Train data contains columns: timestamp, underlying asset value, expiry, call, put strikes, and 41 anonymous features.

Test data contains columns: Timestamp(shuffled and masked), underlying asset value, call, put strikes and 41 anonymous features.

DATASET CLEANING AND PRE-PROCESSING:

```
train_df.describe()
```

	timestamp	underlying	call_iv_23500	call_iv_23600	call_iv_23700	call_iv_23800	call_iv_23900	call_iv_
count	1.783400e+05	178340.000000	178340.000000	1.783400e+05	178340.000000	1.783400e+05	1.783400e+05	178340.0
mean	1.746052e+18	24323.883938	0.423909	-3.220020e+07	0.349236	3.499773e+05	1.429192e+03	1.5
std	3.824884e+14	129.485600	2.178706	6.800128e+09	1.984786	7.389699e+07	5.550653e+05	402.7
min	1.745296e+18	23865.200000	0.186658	-1.435870e+12	0.036630	-1.244180e+05	-9.622650e+04	-0.0
25%	1.745818e+18	24280.600000	0.256790	2.417285e-01	0.226604	2.107440e-01	1.925507e-01	0.1
50%	1.745995e+18	24347.000000	0.297625	2.770240e-01	0.256366	2.354450e-01	2.165680e-01	0.1
75%	1.746432e+18	24385.200000	0.404716	3.753433e-01	0.346050	3.155980e-01	2.813773e-01	0.2
max	1.746610e+18	24668.500000	912.411000	7.904570e+08	833.879000	1.560360e+10	2.334040e+08	120282.0

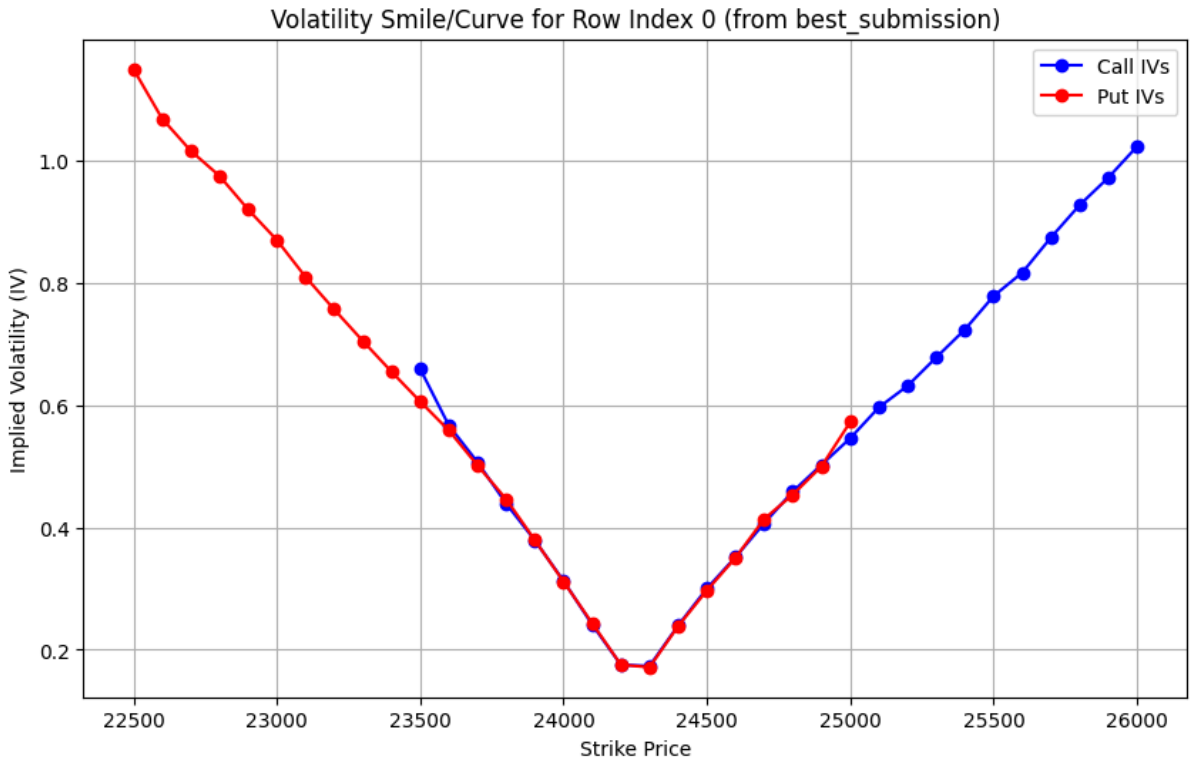
8 rows × 96 columns

- **There are some negative values in the volatility columns, but volatility cannot be negative; therefore, I replaced the values with zeroes.**
- **Also, some volatility values are extremely high in value (probably outliers) which is not possible again, therefore I equated them to the maximum value in that column.**
- **I also made sure that there are no null values in train data.**
- **I scaled all the feature columns using StandardScaler from sklearn.preprocessing.**
- **I repeated the similar steps for test data, checked for any negative values and replaced (if any), checked for outliers in each column, and scaled all feature columns of test data**

PLOTTING THE TRAIN DATA:

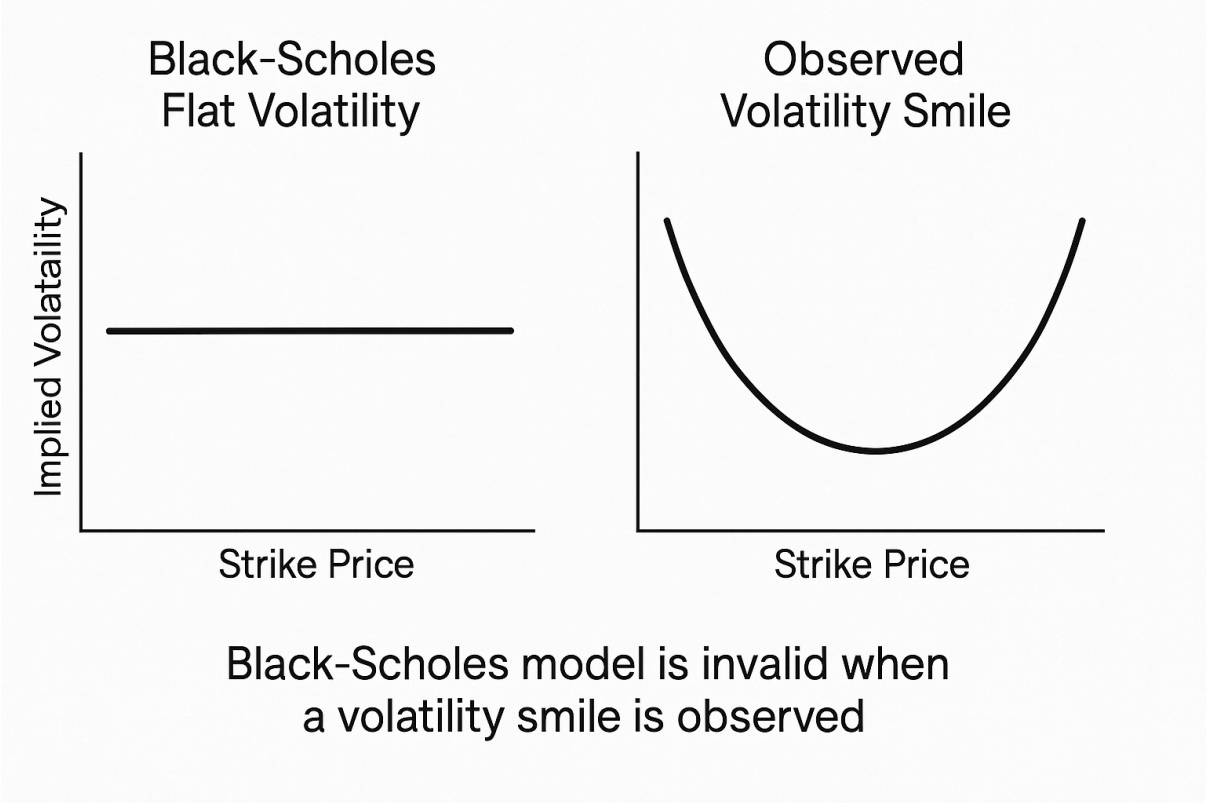
I plotted the values of the train data row-wise and column-wise, and here are the results I got:

Row-wise:

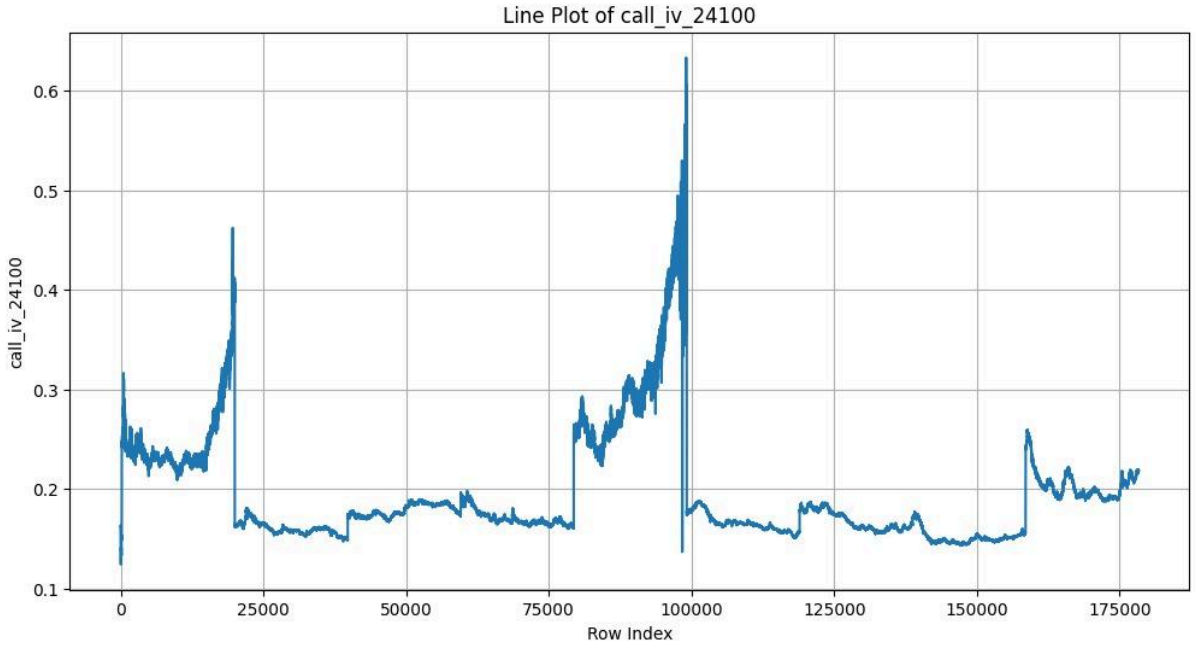


Clearly, the volatility values are following a smile shape/ U-shape, The Black-Scholes model assumes that the volatility of the underlying asset is constant over time and across all strike prices, so this is a real market scenario where black-scholes assumptions aren't true. The diagram below (taken from the internet) shows this pictorially.

INFERENCE FROM ROW PLOTTING:

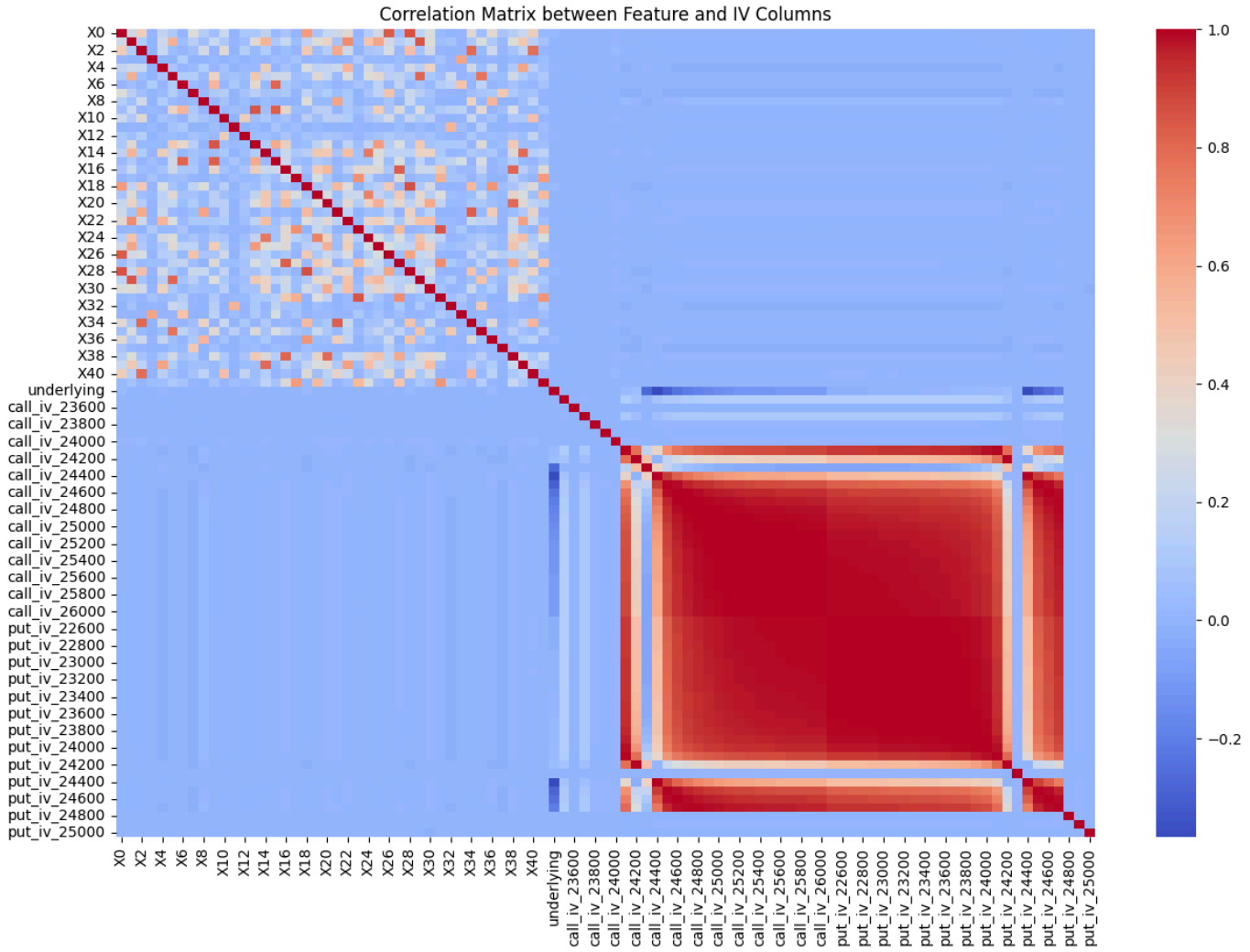


Column-wise:



There is no proper pattern observed when the train data is plotted column-wise, here above is one of the result.

Correlation matrix:



DEVELOPING THE PREDICTIVE MODEL:

As it is evident from the Exploratory data analysis, there is **no linear correlation** between the **feature columns** and the **target columns**, there might be a non-linear or **complex relationship** between these tabular formatted values, therefore its a good sign for us to choose **Random forest model or an XG Boost model**, I cannot go for neural network because training with 1lakh+ samples will take lot of time which is not favourable in trading environments, I cannot use deep learning models related to time series like RNN/LSTM etc because the timestamps are masked and shuffled in test data, so it's **better to go with tree**

based ensembled methods (an advantage of these models are they are also robust to missing values and outliers).

But, as we saw that the volatility follows a smile / U-shaped curve, we might just need to build a model that best predicts the exact curve on the test data. So I can also try to fit cubic/quadratic splines to predict the volatility smile.

So, I tried out different tree based methods based on the above inferences from train data, like Random forest, XGBoost and similar methods like KNN, I also tried spline fitting and smoothing where there is a huge change in derivative of the curve, out of these i achieved best loss with Random forest regressor imputer, To enhance, I picked up **extra tree regressor model as imputer**, I ran it iteratively on test data, and that gave me the best loss out of all models. I chosed **Extra Tree regressors** because they have **high tree diversity** (more accurate results expected), they use **random splits** during split selection whereas Random forest uses best split for split selection, so they are **more accurate and faster** for imputation tasks. **I achieved rank 15, and my best private score of 0.000001126 with extra tree regressor imputer model. (I didn't even use the given 25% values from test data).**

THANK YOU FOR YOUR TIME!

REGARDS
VAMSHI KRISHNA EMMADI
2025 UNDERGRAD, IIT KHARAGPUR