Options Pricing Using Machine Learning

Group 34

Aditya Dhanotia - 9427142877
Jash Patani - 4558521556
Soumya Nambi Ganesh - 6494399900
Shubhangi Sharma - 5801432713
Shantanu Trivikram - 1214307012
Tanvi Vijay - 3001184782

Contact Person
Jash Patani
patani@usc.edu

Executive Summary

The project involves building statistical/ML models to value European call options on the S&P 500 without using the Black-Scholes formula. The training dataset has 1680 separate options with 6 columns. The fields include - current option value, current asset value, strike price of the option, annual interest rate, time to maturity (in years), and result of the Black-Scholes formula applied to this data. The response variables to be predicted using the models are the current option value (regression problem) and whether the Black-Scholes formula overestimates or underestimates the option value (classification problem). The exploratory data analysis showed that there was no significant correlation between any of the predictor variables, making them a good mix of variables to predict the two fields. However, Value had a strong negative correlation with the Strike Price. Next, outlier detection was performed to ensure the quality of the data was good. Data cleaning involved imputing missing values using K-nearest neighbors imputer and filtering out null values that did not carry any valuable information.

For classification modeling, three classifiers, namely, logistic regression, KNN classifier, and random forest classifier (best model), produced the best results. This model was used to classify whether the prediction overestimated or underestimated the option value - a binary result. The random forest classifier outperformed the other classifiers, with an accuracy of 94% and approximately 98.2% precision. For regression, the three models that produced the best results were Linear Regression, KNN Regressor and Random Forest (best model with accuracy of 99%). This model was used to predict the current option value (C).Random Forest was chosen for both processes not only because of the high accuracy results of our models, but also because it accounts for randomness (through sampling, bootstrapping and in choosing predictors at each split). It is also an ensemble methodology and focuses on interpretability.

The trade-off between interpretability and accuracy in machine learning is discussed, where we specify that we are focussing on accuracy. When it comes to outperforming the traditional Black-Scholes model for predicting option values, machine learning models are particularly well-suited to capturing non-linear patterns and relationships between different variables that affect option prices. To predict Tesla's option value, we would need to incorporate so-called “Asymmetric Variables" that are specific to Tesla's stock, such as variables for compound annual growth rate and cash flow projection, as well as Tweets made by Elon Musk which can influence Tesla's stock behavior. Incorporating these variables can create a more accurate model for predicting the value of Tesla's options.

Introduction

Option pricing is a crucial component of financial markets, where investors require accurate predictions of the value of their investment. The Black-Scholes model, developed in 1973, has been a cornerstone of option pricing for decades, providing an analytical framework to calculate the fair value of options. However, this formula has its limitations, such as assuming that the asset prices follow a log-normal distribution and the volatility is constant over time.

With the advent of ML, we have access to powerful algorithms that can learn complex patterns and relationships from data without the need for explicit mathematical models. The benefits of using ML for option pricing are numerous. For example, ML models can capture more complex relationships between the input variables and the option prices, such as non-linear and time-

varying relationships. ML models can also adapt to changing market conditions and incorporate new information as it becomes available.In this project, we will explore the use of various ML models to price European call options on the S&P 500 index.

Problem Statement

The goal of this project is to build regression and classification models to predict the option value and Black-Scholes formula output using the given predictors. The training data set includes information on 1,680 separate options, including the current asset value, strike price of option, annual interest rate, time to maturity, the Black-Scholes formula output, and actual option value. The task is to predict the actual option value (regression problem) and whether the Black-Scholes formula overestimates or underestimates the option value (classification problem). The best-performing model will be used to predict the option value and Black-Scholes formula output for the 1,120 options in the test data set, and its accuracy will be compared with actual results. The objective is to identify an accurate model that can accurately predict the option value and Black-Scholes formula output, which can be used by financial analysts and investors to make informed decisions in the financial markets.

Exploratory Data Analysis

1. The training data had information on 1,680 separate options with 6 columns. In particular, for each option we have recorded the variables as follows:

- Value (C): Current option value
- S: Current asset value
- K: Strike price of option
- r: Annual interest rate
- tau: Time to maturity (in years)
- BS: The Black-Scholes formula was applied to this data to get C_pred and if an option has C_pred – C > 0, i.e., the prediction over-estimated the option value, we associate that option by (Over); otherwise, we associate that option with (Under).

Here the response variables to be predicted using the models are as follows:
1) Value (i.e., a regression problem)
2) BS (i.e., a classification problem)

2. The correlation matrix of the numerical data showed no significant correlation among any of the predictor variables hence all variables can be retained for modeling. It is important to acknowledge that Value has a strong negative correlation with the Strike Price.

Outlier Detection

Data cleaning is a step in the data preprocessing pipeline to ensure the quality of the data. This step is crucial to ensure our model is run on the correct and proper dataset to prevent the prediction of erroneous values. Our dataset contains six columns namely Value (C), Current Asset Value (S), Strike price of the option(K), Annual interest rate (r), Time to maturity in years (tau), Black Scholes formula applied to this data (BS). After the initial data exploration process, we found the existence of null values in some of the columns. Our data cleaning process is essentially done in 3 branches.

1. Identifying records where the Value (C) is Null.
2. The data exploration process revealed the existence of some outliers in these columns.
3. The third step of our outlier detection algorithm defines outliers based on the definition of the variables in the dataset.

Data Imputation

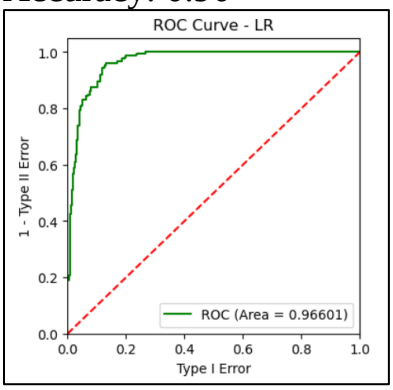
Once we have identified the outliers, we must have an algorithm to clean/impute values in it because we cannot lose valuable data over an outlier in one variable.

The records detected by the first branch of our algorithm (Multiple null values) have to be filtered out as these records do not carry any valuable information and imputing values will introduce more randomness. But for the outliers detected by the next two branches, we need a method to impute values. We use the K-nearest neighbors (KNN) imputer to impute values in these outliers. Once the data cleaning process, which involves outlier detection and data imputation is completed, we can be assured that the data is ready for our regression and classification models. We can trust the prediction and classification accuracy to a reasonable degree.

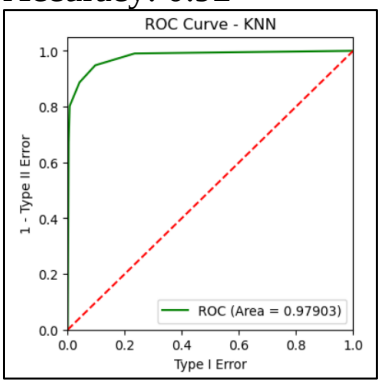
Classification on Options Dataset

Three classifiers among all we used, gave us the best results, Logistic Regression, KNN Classifier and Random Forest Classifier (Best Model) to classify whether our prediction over estimated the option value (Over) or underestimated the option value.

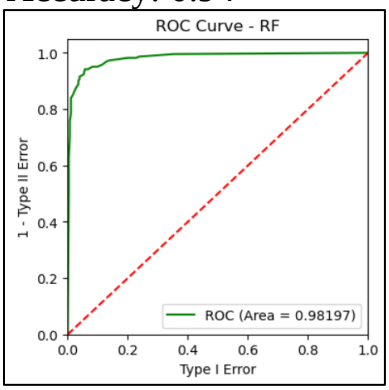
Accuracy: 0.90



Accuracy: 0.92



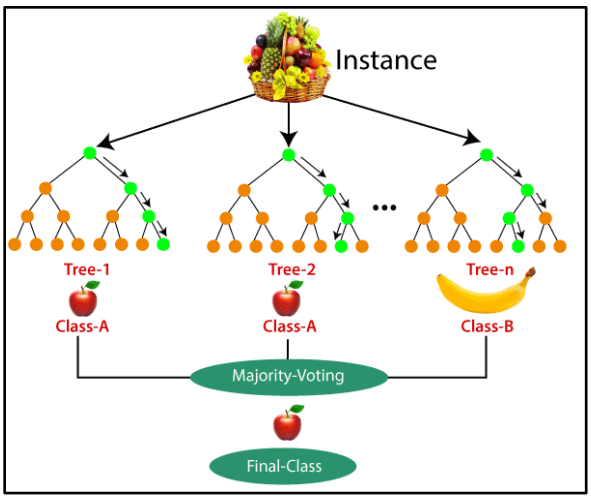
Accuracy: 0.94



We see that Random Forest Classifier outperforms the other classifiers on our dataset among other classifiers. We got the highest accuracy of 94% with approximately AUC 98.2%. Please refer to the appendix for details on the model hyperparameters.

Why Random Forest ?

Random Forests are based on **Ensemble Learning** methodology. Ensemble Learning is nothing but usage of many combined models instead of one. In this case, many decision trees.



Random Forest is based on Bagging Concept where every time while creating a tree, it creates a different training subset from the sample training data with replacement. Each Decision Tree gives an output and final output is based on majority Voting.

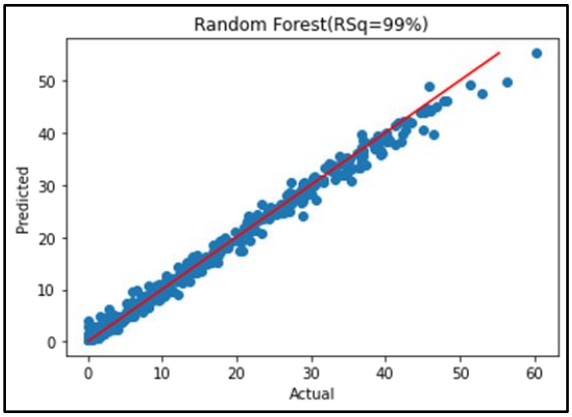
We have summarized the number of correct and incorrect predictions made by our classifier in comparison with the actual class labels and shown in the following confusion matrix.

	Predicted Under	Predicted Over
Actual Under	270 (TN)	11 (FN)
Actual Over	19 (FP)	202 (TP)

From the above confusion matrix, we see that our mis-classification rate is around 6%.

Regression on Options Dataset

While trying to predict Value (C), among the models run, 3 gave us the best results - Linear Regression, KNN, Random Forest.
This model is used to predict current option value and Random Forest was chosen as our final model, not only because of the high accuracy results, but also because RandomForest accounts for randomness, model interpretability and ensemble methodology, as mentioned in the previous section. Please refer to the appendix for details on the model hyperparameters.



Business Understanding and Conclusion

Interpretability versus Accuracy:

Interpretability versus accuracy is a common trade-off in machine learning. In the context of predicting option values, we conducted exploratory data analysis (EDA) and found that all four variables (current asset value, strike price, current interest rate, and time to maturity) play a significant role in predicting option value. As such our model had a high interpretability score. Therefore, we focused on model accuracy and chose the model which would provide the highest accuracy.

What ML model might outperform traditional Black Scholes model:

When it comes to outperforming the traditional Black-Scholes model, machine learning models are particularly well-suited to capturing non-linear patterns and relationships between different variables that affect option prices. Unlike the Black-Scholes model, which assumes a log-normal distribution of asset prices with constant volatility, machine learning models can incorporate more complex patterns that reflect the dynamic and ever-changing nature of financial markets.

Why should we use a different model to predict Tesla's option value:

In order to predict Tesla's option value, we would need to incorporate so-called "Asymmetric Variables'' that are specific to Tesla's stock. To illustrate we would incorporate variables:

1.Variable signifying the compound annual growth rate of tesla. For example, a Price-to-earnings or Price/earnings-to-growth variable
2.Variable for cash flow projection out 5-10 years to capture expected future earnings of the company
3. We may also include Tweets made by Elon Musk as they have a high influence on how Tesla' s stock behaves on a daily basis.

By incorporating these asymmetrical variables, we can create a more accurate model for predicting the value of Tesla's options.

Technical Appendix

EDA

The summary statistics for the variables were as follows:

	Value	S	K	tau	r
count	1678.000000	1679.000000	1678.000000	1679.000000	1680.000000
mean	15.068709	464.402535	438.241955	0.437519	0.030235
std	14.040023	973.652179	23.408989	7.057555	0.000557
min	0.125000	0.000000	375.000000	0.003968	0.029510
25%	2.255001	433.863864	420.000000	0.119048	0.029820
50%	11.190967	442.634081	440.000000	0.202381	0.030130
75%	25.747434	447.320414	455.000000	0.285714	0.030540
max	60.149367	40333.000000	500.000000	250.000000	0.031880

Correlation Matrix of All Fields:

	Value	S	K	tau	r
Value	1.00	-0.01	-0.89	-0.03	0.04
S	-0.01	1.00	-0.01	-0.00	0.05
K	-0.89	-0.01	1.00	0.04	-0.17
tau	-0.03	-0.00	0.04	1.00	-0.01
r	0.04	0.05	-0.17	-0.01	1.00

Details on Cleaning

1. Value is our dependent variable and there is no point imputing values here for us to train the model on. Also, there were two records where the S, k, tau is Null. Imputing values in these cells using any algorithm will introduce a huge amount of randomness.

2. The Current Asset value (S) is greater than 40000, while the mean of all the other records is around 500. This is clearly an outlier, and we must deal with the record in question.

3. Similarly, we see there exists records with time to maturity values in 100's of years which is rationally not possible. To identify these kinds of records, we devised an algorithm which

defines any record value which lies more than 3 standard deviations away from the mean of that column as an outlier. We do that to generalize our outlier detection model and identify records which are clearly an outlier. One caveat to this approach is that it is insensitive to columns with small means and smaller variances. Our outlier detection algorithm detects 3 records with outliers like this.

4. Some examples of these might include:

a. Current Asset value can only be positive i.e greater than zero
b. Time to maturity can only lie in a defined positive range like 0-30 years.
c. Strike price of the option can only be positive

The above is just a non-exhaustive list of filters you can have to define an outlier. The above branch of our algorithm detects 1 record with outliers.

KNN Imputer

The crux of this imputer is that it looks at all the variables of all records in the database. It finds the nearest neighbors based on the values of these variables and assumes that the record in question is similar to its neighbors. It imputes the mean of the outlier variable of the neighbors and imputes the value instead of the outlier.

Modeling

1. Classification

Modelling
We have used the Random Forest Classifier with n_estimators = 30 i.e. There are 30 trees being used to build the model. We have split the data into training and testing in the ratio 80:20 and used the test set to make the predictions. Code snippet of the Random Forest Classifier is as shown below:

```
clf_rf = RandomForestClassifier(random_state=1, n_estimators = 30)
clf_rf.fit(X_train_cl, y_train_cl)
y_pred_rf = clf_rf.predict(X_test_cl)
score_test_rf = accuracy_score(y_test_cl, y_pred_rf)
print("The classification Accuracy is {}".format(round(score_test_rf,2)))
```

```
The classification Accuracy is 0.94
```

2. **Regression**

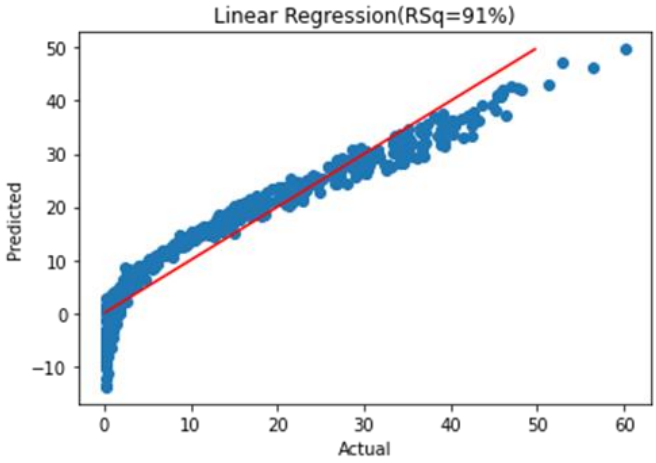
Modelling

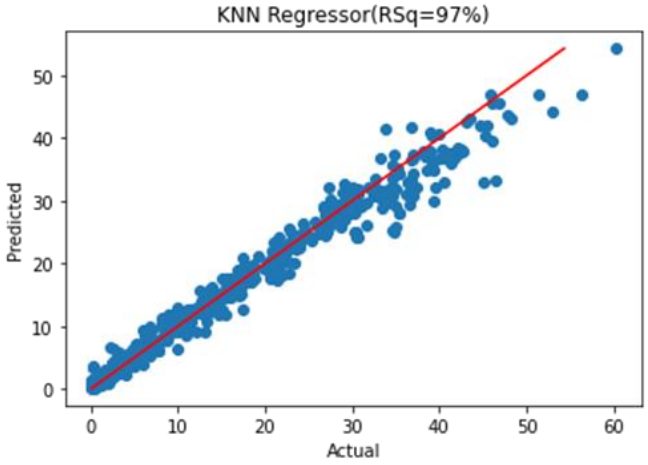
We have used the Random Forest Regressor with n_estimators = 30 number of decision trees being used), with a mx_depth of 10, and max_features of (square root of available features = 2) for each split.

We have split the data into training and testing in the ratio 80:20 and used the test set to make the predictions. Code snippet of the Random Forest Classifier is as shown below:

```
RandomForestRegressor(n_estimators = 30, max_features = 'sqrt', max_depth = 10, random_state = 10).f
```

Other Regression Models:





Other Models

```
[ ]
##########################################
# Appendix
# Other Models
```

```
[135]
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)

model = LinearRegression()
```

✓ 0.0s

```
[85]
model.fit(X_train, y_train)
in_sample_r_square = model.score(X_train, y_train)
out_of_sample_r_square = model.score(X_test, y_test)
```

```
[86]
out_of_sample_r_square
```

```
0.9148114682358571
```

```
[87]
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
```

```
[88]
kfolds_regresssion = KFold(n_splits = 5, random_state = 1, shuffle = True)
regresssion_model = LinearRegression()
r2_model_1_cv = cross_val_score(regresssion_model, X, y, cv=kfolds_regresssion)
```

+ Code + Markdown

```
[89]
np.mean(r2_model_1_cv)
```

```
0.9102287981733787
```

```
[90]
from sklearn.neighbors import KNeighborsRegressor
```

```
[ ]
neigh = KNeighborsRegressor(n_neighbors=8)

neigh.fit(X_train, y_train)
```

```
[93]
neigh.score(X_test, y_test)
```

```
0.9674403343478979
```

KNN Classifier

[99]
```
X = df2
y = data['BS']
```

+ Code + Markdown

[100]
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
```

[103]
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
```

[113]
```
clf = RandomForestClassifier(max_depth=10, random_state=0)
```

[114]
```
clf.fit(X_train, y_train)
```

```
RandomForestClassifier(max_depth=10, random_state=0)
```

[115]
```
clf.score(X_test, y_test)
```

```
0.9404761904761905
```