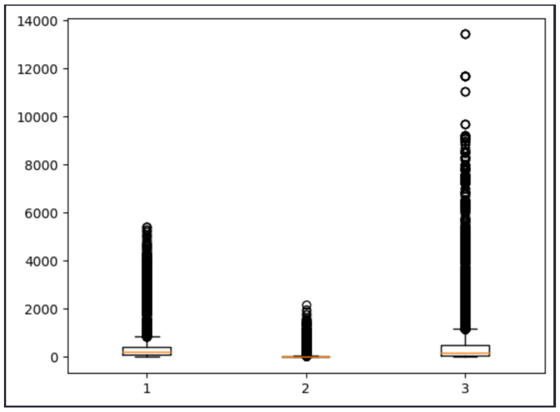
## Prompt

Given an options strategy dataset, it prompts you to report on any patterns you identify that may be used to (1)Increase fill rate,  and/or reduce latency.

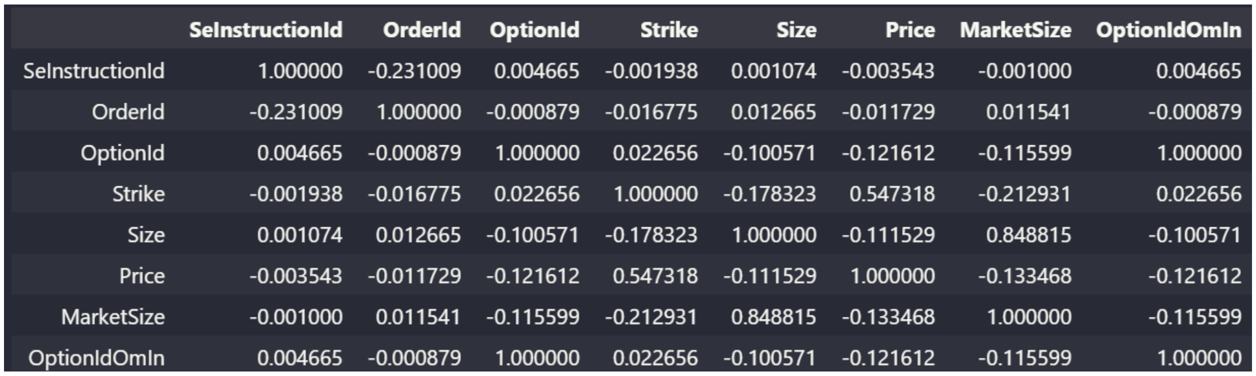
## Basic understanding and preparing

Options are contracts that allow the right to buy or sell stock at an agreed piece(Strike), on or before a particular date(Expiration). At the same time, options can be traded as well(price). Based on the prompt, I deduce I am asked for a pattern that could influence fill rate and latency. From a statistical perspective, it is asking for what indicators in the pattern might influence fill rate or latency. After understanding the prompt, I am about to generate one hypothesis or several ones, using statistical methods to validate or prove my hypothesis wrong. From there, do testing on data to refine my results

## Data cleaning and Exploration



Since the concern is increasing the fill rate of the option orders, I think I don't need a glance at those canceled orders. That's why I dropped all Types of "REJECT" and empty data rows if any. First I made a general look at the data set with boxplots on potentially important numerical indicators:
"Strike","Price", "MarketSize". Later I will check all of the indicators' correlations to gain a basic understanding of indicators correlations to help me make an appropriate hypothesis.



| | SeInstructionId | OrderId | OptionId | Strike | Size | Price | MarketSize | OptionIdOmIn |
|---|---|---|---|---|---|---|---|---|
| SeInstructionId | 1.000000 | -0.231009 | 0.004665 | -0.001938 | 0.001074 | -0.003543 | -0.001000 | 0.004665 |
| OrderId | -0.231009 | 1.000000 | -0.000879 | -0.016775 | 0.012665 | -0.011729 | 0.011541 | -0.000879 |
| OptionId | 0.004665 | -0.000879 | 1.000000 | 0.022656 | -0.100571 | -0.121612 | -0.115599 | 1.000000 |
| Strike | -0.001938 | -0.016775 | 0.022656 | 1.000000 | -0.178323 | 0.547318 | -0.212931 | 0.022656 |
| Size | 0.001074 | 0.012665 | -0.100571 | -0.178323 | 1.000000 | -0.111529 | 0.848815 | -0.100571 |
| Price | -0.003543 | -0.011729 | -0.121612 | 0.547318 | -0.111529 | 1.000000 | -0.133468 | -0.121612 |
| MarketSize | -0.001000 | 0.011541 | -0.115599 | -0.212931 | 0.848815 | -0.133468 | 1.000000 | -0.115599 |
| OptionIdOmIn | 0.004665 | -0.000879 | 1.000000 | 0.022656 | -0.100571 | -0.121612 | -0.115599 | 1.000000 |

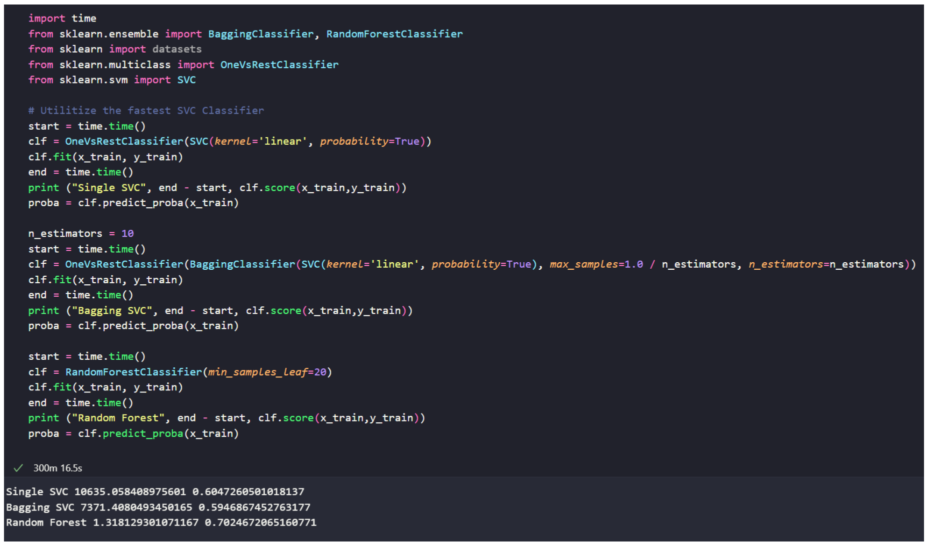
The boxplot is nearly gradual for “Strike” and “Price”. For the boxplot, there are quite a lot of outliers in the data set. In some sense, these outliers may no longer be the typical “outliers”, indicating the spread of strike, price, and market size is massive, this is one of the important patterns of this data set. For correlations, there are two relatively strong correlations I found, the first pair is “Strike” and “Price”(correlation coefficient = 0.547318), the second pair is “Market Size and Size”(correlation coefficient = 0.848815), given the data.  In this data set, “Strike” or “Price” hardly tell anything about “MarketSzie” nor “Size”, as the correlations between them are very close to 0.

## Hypothesis

I think important indicators may include “Strike”, “Price”, “MarketSize” and “OrderEventTypeOmIn”. First of all, I don’t think any of the indicators can imply a change in latency. Latency is determined mainly by the speed of order execution, how fast the interaction is between market data and exchange, and hardware and software performance. None of the indicators given can give a direct hint about latency in this scenario. It is somehow hard to determine whether “Strike” influences fill rate as limited information is given, the “real asset” price, I can’t see the difference between the strike price and real asset price, thus I cannot determine the liquidity of one particular option order for sure. A lower price of an order does show the liquidity of the option ideally, as most investors can afford. If the size of an option is large, especially given that the market size is relatively small, it will ideally take longer for orders to be filled. Bigger market size theoretically implies higher liquidity and faster fill rates as the number of contracts is more in amounts. It is hypothesized that an option order has a lower price, large market size and relatively small size gains a faster fill rate.

# Test hypothesis

Given the dataset with the key variable, “**OrderEventTypeOmIn**”. I have removed the “REJECT” type as they provide no hint of whether an order is filled or not. Noticing I have two classifications right here. Based on the prompt I can deduce that we care about how these indicators I mentioned above contribute to the likelihood of an order being “FILL” or “CANCEL”. Mapping multiple indicators to a high dimensional feature space so that all the data points I am concerned about can be categorized, in this case as (“FILL” or “CANCEL”). I decided to use the SVM (Support Vector Machine)algorithm to see which indicator is more significant, then show the pattern. As I am given the classification type, it is appropriate to use a supervised machine learning algorithm. I once applied the algorithm, but somehow it’s not working out. After researching online resources, I found out that my y values(“FILL” and “CANCEL”) are both string literal, so I transformed the string literals in column “OrderEventTypeOmIn” into numerical values for later operation. Randomly choose 80% of the data for training my model and 20% of the data to validate to what extent my hypothesis is accurate. More importantly, the kernel function is very useful when describing the non-linear observations, here I got several observations as my independent variable x, and my dependent variable as “OrderEventTypeOmIn”. For the exact implementation, please see my jupyter notebook attached. In the process of implementing the SVM model, I tried to spend one night training my model, the program was still running when I woke up in the morning. From then on, I decided to apply some utilized classifiers to boost my training. Thus I trained my data and I chose the most reliable classifier with the highest score.

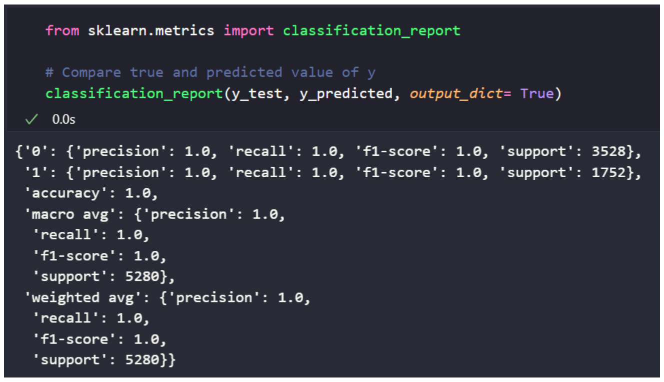


```python
import time
from sklearn.ensemble import BaggingClassifier, RandomForestClassifier
from sklearn import datasets
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC

# Utilitize the fastest SVC Classifier
start = time.time()
clf = OneVsRestClassifier(SVC(kernel='linear', probability=True))
clf.fit(x_train, y_train)
end = time.time()
print ("Single SVC", end - start, clf.score(x_train,y_train))
proba = clf.predict_proba(x_train)

n_estimators = 10
start = time.time()
clf = OneVsRestClassifier(BaggingClassifier(SVC(kernel='linear', probability=True), max_samples=1.0 / n_estimators, n_estimators=n_estimators))
clf.fit(x_train, y_train)
end = time.time()
print ("Bagging SVC", end - start, clf.score(x_train,y_train))
proba = clf.predict_proba(x_train)

start = time.time()
clf = RandomForestClassifier(min_samples_leaf=20)
clf.fit(x_train, y_train)
end = time.time()
print ("Random Forest", end - start, clf.score(x_train,y_train))
proba = clf.predict_proba(x_train)
```

✓ 300m 16.5s

```
Single SVC 10635.058408975601 0.6047260501018137
Bagging SVC 7371.4080493450165 0.5946867452763177
Random Forest 1.318129301071167 0.7024672065160771
```

I used my 20% test data to test the model trained. It turned out to be very reliable, making all predictions correct (thanks to the massive load of data)



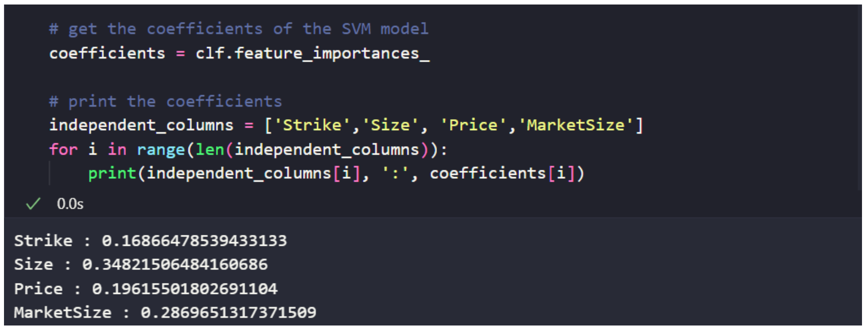
```python
from sklearn.metrics import classification_report

# Compare true and predicted value of y
classification_report(y_test, y_predicted, output_dict= True)
```

✓ 0.0s

```
{'0': {'precision': 1.0, 'recall': 1.0, 'f1-score': 1.0, 'support': 3528},
 '1': {'precision': 1.0, 'recall': 1.0, 'f1-score': 1.0, 'support': 1752},
 'accuracy': 1.0,
 'macro avg': {'precision': 1.0,
  'recall': 1.0,
  'f1-score': 1.0,
  'support': 5280},
 'weighted avg': {'precision': 1.0,
  'recall': 1.0,
  'f1-score': 1.0,
  'support': 5280}}
```

Based on the features importance report. It finalizes my conclusion.

## Conclusion



```
# get the coefficients of the SVM model
coefficients = clf.feature_importances_

# print the coefficients
independent_columns = ['Strike','Size', 'Price','MarketSize']
for i in range(len(independent_columns)):
    print(independent_columns[i], ':', coefficients[i])
```

✓ 0.0s

```
Strike : 0.16866478539433133
Size : 0.34821506484160686
Price : 0.19615501802691104
MarketSize : 0.2869651317371509
```

Recall the hypothesis I made: “It is hypothesized that an option order has a lower price, large market size, and relatively small size gains a faster fill rate.” What I did can hardly tell whether one unit change of one specific independent variable caused unit changes of the dependent variable. While it is safe to conclude that “Size” is the most important feature of all these related independent variables. It is mentioned in the hypothesis section that Strike and Price may be hard to predict whether the order is filled or not. Overall, “Size” and “MarketSize ” have relatively more feature importance when talking about whether an order is going to be filled or not. This could be very useful to take more look at “MarketSize” and “Size” when trading option orders as it may imply your option order is easier to get transacted.

## Reflection and Improvement

I made a reasonable hypothesis, but what I’ve done is kind of a leap but not a complete one to where I want. SVM does not have real coefficients that can be easily interpreted. While it shows to what extent the independent variable is important. Since it is concluded that the “Size” variable is the most important variable for predicting the fill rate. I know it's important to continue iterating and refining my analysis to uncover new insights and validate your hypotheses.  I will continue to do research on this project, trying to go in depth to figure out the causation relationship of these variables as possible as I am capable of.