DSO530: Applied Modern Statistical Learning Methods

Option Pricing Model Project

Professor. Xin Tong
Spring 2023

Team 21

Thong (Tom) Do
Qiao (Jessie) Xue
Connie Yau
Huiling Xiao
Jiayi (Lily) Hu
Yanshuo Li

USC Marshall
Master of Science in Business Analytics

Executive Summary

Our goal for this report is to develop machine learning models that accurately predict option prices and determine whether using the Black-Scholes algorithm will underestimate or overestimate the actual price. To achieve this, we first cleaned data by removing outliers beyond 3 standard deviations from the mean and dropping records with null values. We then introduced two new features: the Future Value of an Investment and the Intrinsic Value of a Call Option, with the aim of improving model performance.

We explored various models, starting with linear and logistic regression as our baseline, followed by non-linear models such as decision trees, random forests, XGBoost, LightGBM, and Gradient Boosting. We tuned each model's hyperparameters to achieve optimal performance. After all these, we splitted the train and test set, did 5-fold cross-validation on randomly shuffled dataset, and repeated this process 100 times to evaluate the model and pick the best one.

Our approach revealed that dropping outliers and adding new features improved model performance. For example, linear regression's adjusted R-square increased from 0.797 to 0.983. We also found that strike price, future_mult, and S/K were highly correlated with option price. After running all models, we concluded that random forest, XGBoost, and Gradient Boosting were the best regression models, while non-linear models outperformed logistic regression for classification with a minor issue of overfitting.

In the end, we opted for Gradient Boosting with n_estimators=100, learning_rate=0.1, and max_depth=4 as our final regression model for several reasons. Firstly, it has high in-sample, out-of-sample, and cross validation R-squared values with low deviation. In fact, its testing R-squares were above 0.9985 93% of the time. Additionally, it offers some interpretability with feature importance scores. Finally, it is fast and scalable, making it suitable for large datasets, which is important for quantitative finance modeling.

For classification, we selected CatBoostClassifier with iterations=75, depth=10, learning_rate=0.1, l2_leaf_reg=25, and loss_function='Logloss'. This model stands out due to its comparatively high in-sample, out-of-sample, and cross validation accuracy. It can achieve above 90% accuracy 96% of the time and above 92% accuracy 72% of the time on testing, which meets our project requirements. Additionally, it runs faster during training, which is essential when working with large datasets.

In summary, our machine learning model accurately predicted option prices. Our chosen regression and classification models demonstrated high performance, interpretability, and scalability for large datasets, meeting the requirements for quantitative finance modeling.

Analysis Approaches

1. Data Exploration

During the data exploration phase of our study, we closely examined a dataset composed of five quantitative variables (Value, S, K, tau, and r) associated with European-style option pricing. This dataset encompasses 1,680 records spread across five distinct fields. Our analysis revealed that the dataset possesses minimal missing data, with each variable maintaining a population rate of at least 99.88% (see Appendix 1). Additionally, our univariate analysis uncovered a diverse array of values for each variable, and some demonstrated considerable variability, as highlighted by their respective standard deviations (see Appendix 1). We also identified several outliers and anomalies in the distribution of S and tau (see Appendix 2), which we plan to address and rectify during the data cleaning stage.

2. Data Cleaning

In order to ensure the integrity of our dataset, we performed two primary data cleaning tasks. Firstly, we eliminated records containing null values to maintain consistency and accuracy throughout the dataset.

Row Index	Value	S	K	tau	r	BS	
12	2.315001441	448.6881092	470	250	0.03013	Over	Outliers
33	2.565000016	445.04224	455	146	0.03003	Over	
47	11.45127253	40333	425	0.043650794	0.03147	Under	
879	4.125	0	455	0.170634921	0.03003	Over	
53	NAN	446.7189744	430	0.166666667	0.02962	Under	Missing Values
292	8.624999997	NAN	NAN	NAN	0.03003	Over	
818	NAN	431.2846156	NAN	0.23015873	0.02972	Over	

Secondly, we identified and removed outliers that fell beyond three standard deviations from the mean of each predictor, as these values may skew the analysis and lead to erroneous conclusions. Dropping outliers boosted the Linear Regression R-square from 0.798 to 0.912 (see Appendix 3).

3. Features Engineering

Beyond 4 provided features (S: current price, K: strike price, tau: time to maturity, and r: annual interest rate), we created 2 new features to boost the performance of our model:

Feature name	Method	Description
future_mult	$(1 + r)^{tau}$	**Future value of an investment.** For example, if you have an investment with an annual interest rate of 5% and a time to maturity of 10 years, the expression $(1 + 0.05)^{10}$ is the value of each $1 you invested today after the 10-year investment period.
S/K	S/K	**Ratio of Current Asset Value and Strike Price of Option:** If the result is greater than 1, the option is "in the money" because the option holder could buy the underlying asset at a lower price than the current market price. Vice Versa. For example, if a call option has a K = $100 and S = $120, the intrinsic value of the option would be $20. This means the option holder could exercise the option and buy the underlying asset for $100, then sell it on the market for $120, realizing a profit of $20

By adding these 2 new features, we were able to boost the adjusted R-squared of a linear regression model from about 0.912 to around 0.983 (see Appendix 3).

4. Model Exploration

Our objective for model exploration was to experiment with different models to select the best model for both regression and classification problems. In the regression problem, we wanted to train a model that can accurately predict the option price. In the classification problem, we wanted to build a model that can accurately classify whether using the Black-Scholes algorithm would underestimate or overestimate the actual option price.

Our strategy was to first start with linear regression for the regression problem and logistic regression for the classification problem as the baseline. Linear regression and logistic regression are simpler and more interpretable models, providing quick insights of how input features affect the predicting outcome or target variable.

Then, we experimented with more non-linear models such as decision tree, random forest, and boosting models, tuning different hyperparameters. To find the best set of hyperparameters for each type of model, we used 5-fold cross validation with random_state of 1 and selected the one with the highest average cross validation R-squared for regression model and highest average cross validation accuracy for classification model.

Problem	Models	Tuning hyperparameters
Regression and Classification	Decision Tree	max_depth (2-10), min_samples_split (5-50), min_samples_leaf (5-30)
	Random Forest	max_depth (2-10), min_samples_split (5-50), min_samples_leaf (5-30), n_estimators (50-100), n_features (2-5)
	LightGBM	n_estimators (10-100), max_depth (2-8), num_leaves (5-30), learning_rate (0.01/0.1)
	XGBoost	n_estimators (10-100), max_depth (2-10), learning_rate (0.01/0.1), subsample (0.1-0.9), gamma (0.1-0.9)
Regression Only	Linear Regression	Baseline Model
	Gradient Boosting	n_estimators (20-200), learning_rate (0.01/0.1), max_depth (2-8)
Classification Only	Logistic Regression	Baseline Model
	CatBoost	iterations (20-100), depth (3-15), learning_rate (0.01-0.1), l2_leaf_reg (5-50), loss_function = "Logloss"

After obtaining the best set of hyperparameters for each model type, we benchmarked all non-linear models together and with linear regression and logistic regression. Our process was to run random train test split (80/20) and random shuffled 5-fold cross validation on all models and record their training, testing, and cross validation R-square and accuracy for 100 times. This approach allowed us to observe how well and stable each type of model performed on different data scenarios to select the final models. In general, we favor models that have higher scores on training, testing, and cross-validation, while exhibiting less variability. This criterion indicates that the selected model is both highly accurate and robust, which can make more reliable predictions.

5. Final Model Selection

After running each model 100 times, we visualized the performance of each model using a side-by-side boxplot for the regression and classification problems (see Appendix 5). For regression problem, non-linear models outperformed linear regression model. Both in-sample, out-of-sample, and cross validation R-squares of random forest and boosting models are near perfect and show very little deviation. Similarly, for classification problem, non-linear models also outperformed logistics regression model. However, these models had a sign of overfitting as the accuracy on training was quite higher than testing and cross validation, contradicting to the pattern on logistics regression.

For selecting the final models, we mainly focused on two selection criteria. First, the higher performance with a smaller standard deviation within sets is better, especially in cross validation and test sets. Second, the differences in performance between train, CV, and test sets are not too large, which indicates that the model is not overfitting or underfitting. According to these rules, we chose the following models as our final models:

Problem	Final Model	Hyperparameters	Training	Testing	Cross Validation
Regression	Gradient Boosting	n_estimators=100, learning_rate=0.1, max_depth=4	R-square: 0.9996	R-square: 0.9988	R-square: 0.9988
Classification	CatBoost	iterations=75, depth=10, learning_rate=0.1, l2_leaf_reg=25, loss_function='Logloss'	Accuracy: 0.9381	Accuracy: 0.9271	Accuracy: 0.9247

Final Approach

After selecting the best regression and classification model, we retrained the final model using all the observations that we had access to. The training dataset is the one we obtained from the data cleaning step, which is free of missing values and outliers on each independent variable. Removing outliers from the training set can prevent the model from learning from noise. After retraining, the gradient boosting model for regression problem achieved an in-sample R-squared of 0.9999, while the CatBoost model for classification problem achieved a training accuracy of 0.9390.

Before making predictions, we made the 2 new features from the 4 original features on the test set by applying the same calculation from the feature engineering step. This step was crucial, as our trained model needed more than 4 original features.

Conclusions

By conducting data exploration and cleaning, feature engineering, model exploration, and model selection, we created and trained a regression and a classification model to predict the specific option price. However, the stock market is changeable, and each option has its own features. Therefore, we need to be careful when choosing the methodology for option price predictions. Here we discuss the insights of modeling for the option market and the limitations of our models.

In the context of valuing European options, we believe that prediction accuracy should be prioritized over model interpretability. Our model's ability to accurately value asset prices can better inform profitable trading and risk management decisions by ensuring that investors have access to the most on target predictions. Specifically, small errors in model prediction can incur significant losses for investors, which may have not only short-term financial consequences but also long-term repercussions such as damage in reputation. Thus, we see the importance of accuracy in predictions especially for a competitive industry such as finance, where even a small difference can make or break investor performance.

While the Black-Scholes model is one the most important concepts of modern financial theory, it presents some major drawbacks that make machine learning models a better option for valuing assets. In particular, the Black-Scholes model makes unrealistic assumptions such as implied volatility, lognormal stock price distribution, zero transactions costs and taxes, among others. Furthermore, it can only be applied to valuing European options on non-dividend paying stocks. Thus, machine learning models outperform the Black-Scholes model in that machine learning models offer considerably more flexibility of inputs and assumptions, and can account for additional variables through feature engineering.

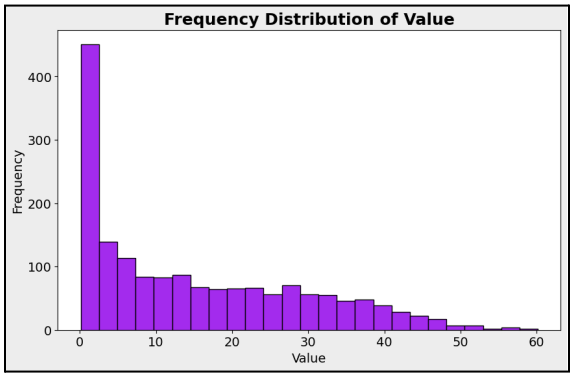
Based on the information on the report, if we had to use our final model to predict Tesla stocks, we will not be comfortable with that for various reasons. Firstly, financial markets are highly complex and influenced by numerous unpredictable factors. The model, which relies on historical data, may not be able to capture the intricate dynamics and real-time changes that significantly impact Tesla's stock price. Therefore, it is suggested to use the model's predictions as a starting point and complement them with additional research and analysis to make well-informed investment decisions. Secondly, the quality of the data used to train and validate the model is crucial for its performance. Before relying on the model to predict option values for Tesla stocks, it is essential to ensure that the training and validation data used are of sufficient quality. However, this is challenging since the details regarding the data provided, particularly if it contains information specific to Tesla stocks, are unknown. Lastly, while we are not comfortable for the reasons above. We believe that if we had Tesla stock data, we would be comfortable using it to train our model and then using it to predict Tesla stock prices.

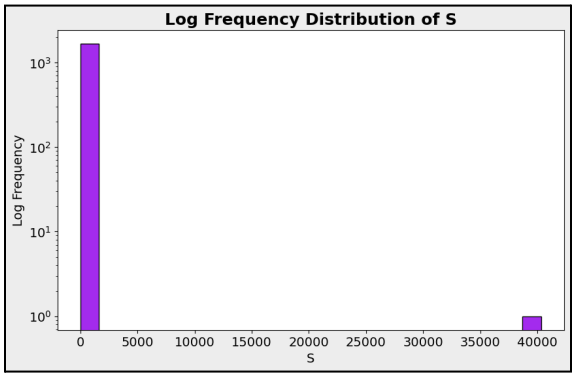
Appendix

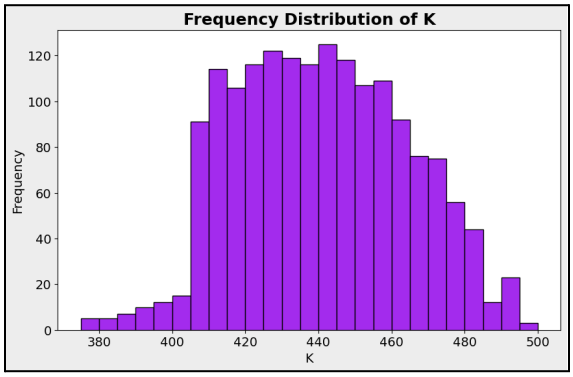
1. Data Summary Table

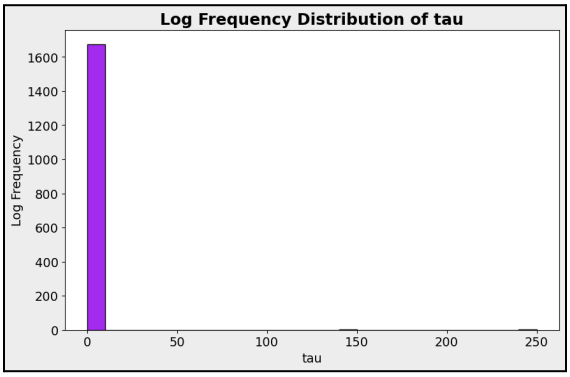
Field Name	% Populated	Min	Max	Mean	Stdev	# Zero
Value	99.88	0.125	60.149367	15.07	14.04	0
S	99.94	0	40333	464.40	973.65	1
K	99.88	375	500	438.24	23.41	0
tau	99.94	0.003968	250	0.44	7.06	0
r	100	0.029510	0.031880	0.03	0.000557	0

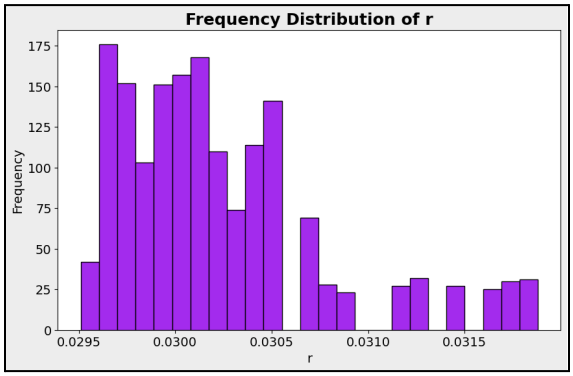
2. Distribution of Each Fields

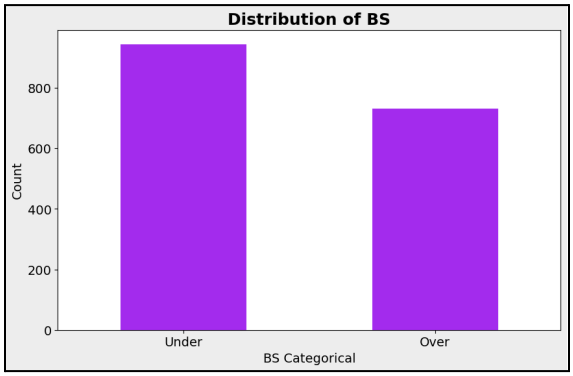












3. Linear Regression Results

Linear Regression (No Drop Outliers + No New Feature)

OLS Regression Results

Dep. Variable:	Value	R-squared:	0.798
Model:	OLS	Adj. R-squared:	0.797
Method:	Least Squares	F-statistic:	1651.
Date:	Tue, 02 May 2023	Prob (F-statistic):	0.00
Time:	21:19:36	Log-Likelihood:	-5469.0
No. Observations:	1677	AIC:	1.095e+04
Df Residuals:	1672	BIC:	1.098e+04
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	341.4870	9.475	36.042	0.000	322.903	360.071
S	-0.0002	0.000	-0.979	0.328	-0.000	0.000
K	-0.5436	0.007	-81.121	0.000	-0.557	-0.530
tau	0.0104	0.022	0.476	0.634	-0.032	0.053
r	-2914.5323	281.588	-10.350	0.000	-3466.835	-2362.230

Omnibus:	26.755	Durbin-Watson:	2.012
Prob(Omnibus):	0.000	Jarque-Bera (JB):	26.710
Skew:	0.287	Prob(JB):	1.58e-06
Kurtosis:	2.769	Cond. No.	2.00e+06

Linear Regression (Drop Outliers + No New Feature)

OLS Regression Results

Dep. Variable:	Value	R-squared:	0.912
Model:	OLS	Adj. R-squared:	0.912
Method:	Least Squares	F-statistic:	4315.
Date:	Tue, 02 May 2023	Prob (F-statistic):	0.00
Time:	21:22:55	Log-Likelihood:	-4762.6
No. Observations:	1673	AIC:	9535.
Df Residuals:	1668	BIC:	9562.
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-22.2504	11.244	-1.979	0.048	-44.304	-0.197
S	0.6216	0.016	39.725	0.000	0.591	0.652
K	-0.5903	0.005	-129.944	0.000	-0.599	-0.581
tau	31.6364	1.048	30.198	0.000	29.582	33.691
r	515.3516	207.201	2.487	0.013	108.949	921.754

Omnibus:	156.358	Durbin-Watson:	2.080
Prob(Omnibus):	0.000	Jarque-Bera (JB):	202.106
Skew:	0.851	Prob(JB):	1.30e-44
Kurtosis:	3.063	Cond. No.	1.26e+06

Linear Regression (Drop Outliers + New Features)

OLS Regression Results

Dep. Variable:	Value	R-squared:	0.983
Model:	OLS	Adj. R-squared:	0.983
Method:	Least Squares	F-statistic:	1.594e+04
Date:	Tue, 02 May 2023	Prob (F-statistic):	0.00
Time:	21:16:59	Log-Likelihood:	-3392.1
No. Observations:	1673	AIC:	6798.
Df Residuals:	1666	BIC:	6836.
Df Model:	6		
Covariance Type:	nonrobust		

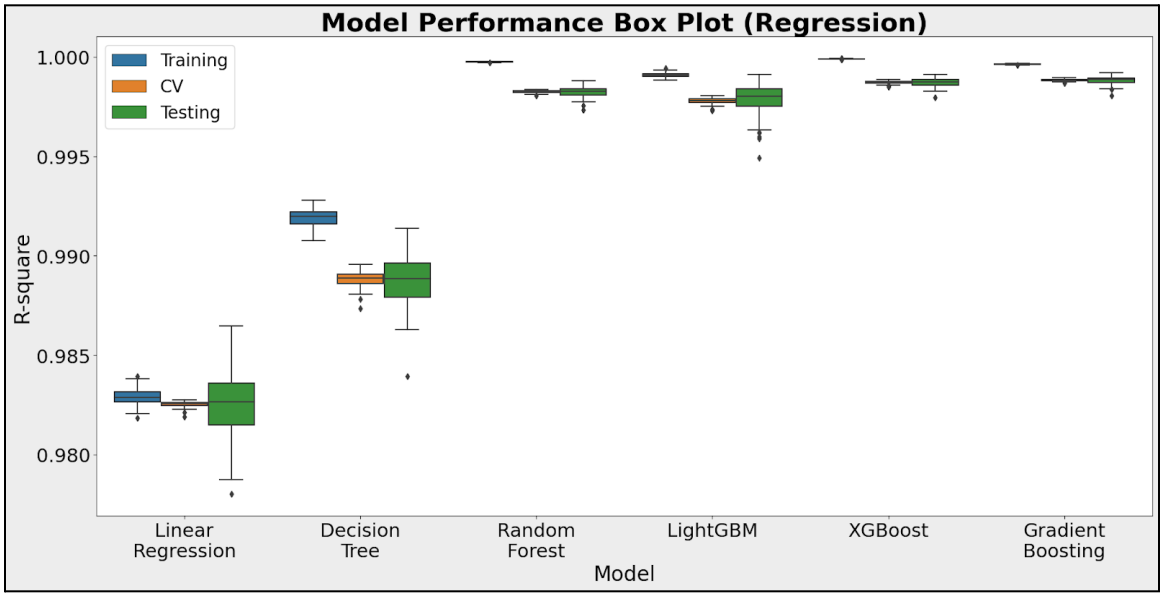
	coef	std err	t	P>|t|	[0.025	0.975]
const	-839.9768	863.510	-0.973	0.331	-2533.656	853.702
S	-2.2513	0.035	-63.781	0.000	-2.321	-2.182
K	2.2951	0.035	65.925	0.000	2.227	2.363
tau	36.1218	26.041	1.387	0.166	-14.955	87.199
r	-34.9968	187.281	-0.187	0.852	-402.329	332.335
future_mult	-428.7784	869.073	-0.493	0.622	-2133.369	1275.813
S/K	1256.0320	15.131	83.012	0.000	1226.355	1285.709

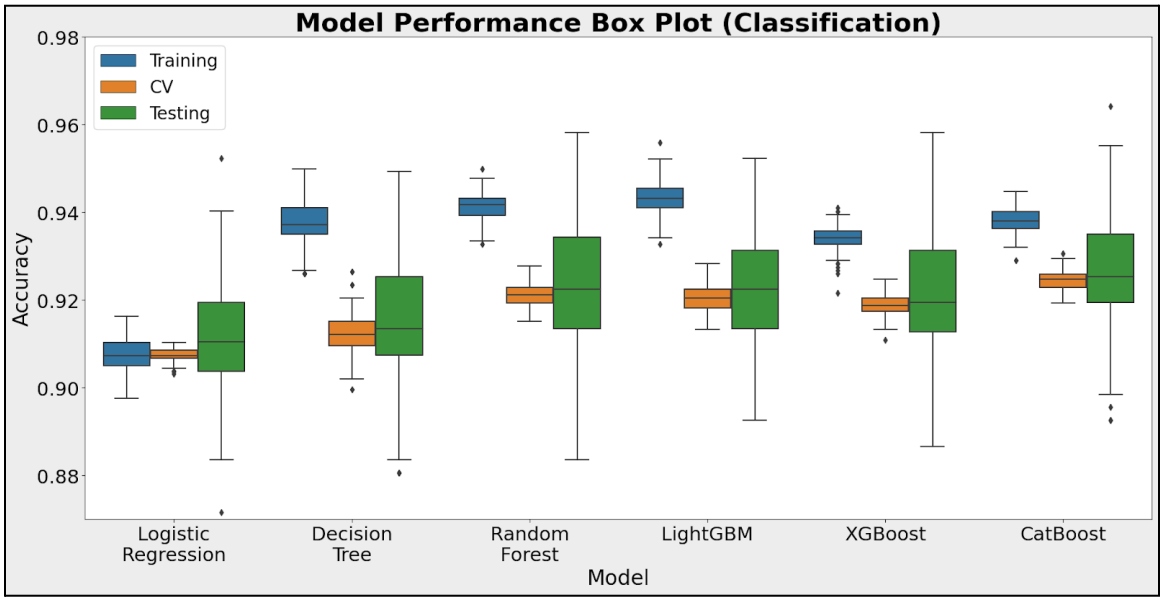
Omnibus:	159.575	Durbin-Watson:	2.092
Prob(Omnibus):	0.000	Jarque-Bera (JB):	852.259
Skew:	-0.266	Prob(JB):	8.60e-186
Kurtosis:	6.456	Cond. No.	1.71e+07

4. Best Tuning Hyperparameters and Performance after 100 Runs

Problem	Final Model	Best Hyperparameters	Mean Training	Mean Testing	Cross Validation
Regression	Gradient Boosting	n_estimators=100, learning_rate=0.1, max_depth=4	R-square: 0.9996	R-square: 0.9988	R-square: 0.9988
	Decicion Tree	max_depth = 10,min_samples_leaf = 20	R-square: 0.9919	R-square: 0.9888	R-square: 0.9888
	Random Forest	Default settings	R-square: 0.9997	R-square: 0.9982	R-square: 0.9982
	Light GBM	Default settings	R-square: 0.9991	R-square: 0.9979	R-square: 0.9978
	XGBoost	n_estimators=100,max_depth = 5	R-square: 0.9999	R-square: 0.9987	R-square: 0.9987
	Linear Regression	Baseline model	R-square: 0.9829	R-square: 0.9826	R-square: 0.9825
Classification	CatBoost	iterations =75, depth=10, learning_rate=0.1, l2_leaf_reg=25, loss_function='Logloss'	Accuracy: 0.9381	Accuracy: 0.9271	Accuracy: 0.9247
	Decicion Tree	max_depth=5, min_samples_split=20, min_samples_leaf =10	Accuracy: 0.9374	Accuracy: 0.9155	Accuracy: 0.9121
	Random Forest	n_estimators=75, max_depth = 5, max_features=5, min_samples_split=25, min_samples_leaf =10	Accuracy: 0.9411	Accuracy: 0.9231	Accuracy: 0.9212
	Light GBM	n_estimators = 12, max_depth = 5, num_leaves = 20, learning_rate = 0.1	Accuracy: 0.9436	Accuracy: 0.9220	Accuracy: 0.9204
	XGBoost	n_estimators = 50, max_depth = 5, learning_rate = 0.01, subsample = 0.25, gamma = 0.3	Accuracy: 0.9338	Accuracy: 0.9213	Accuracy: 0.9188
	Logistic Regression	Baseline model	Accuracy: 0.9075	Accuracy: 0.9099	Accuracy: 0.9074

5. Box-plot Comparing Model Performance





6. Feature Importance Graphs