Pricing with Machine Learning

Splitting the data

8 sets of data in total for training:

- European Options: calls/puts
- American Options: calls/puts

```
X_train_calls, X_test_calls, y_train_calls, y_test_calls = train_test_split(X_calls, y_calls, test_size=0.25)
X_train_puts, X_test_puts, y_train_puts, y_test_puts = train_test_split(X_puts, y_puts, test_size=0.25)

Train calls: (27187, 5)
Test calls: (9063, 5)
Train puts: (18194, 5)
Test puts: (6065, 5)
```

Models

```
XGBr – Xtreme Gradient Booster
DTR – Decision Tree Regressor
```

XGBr

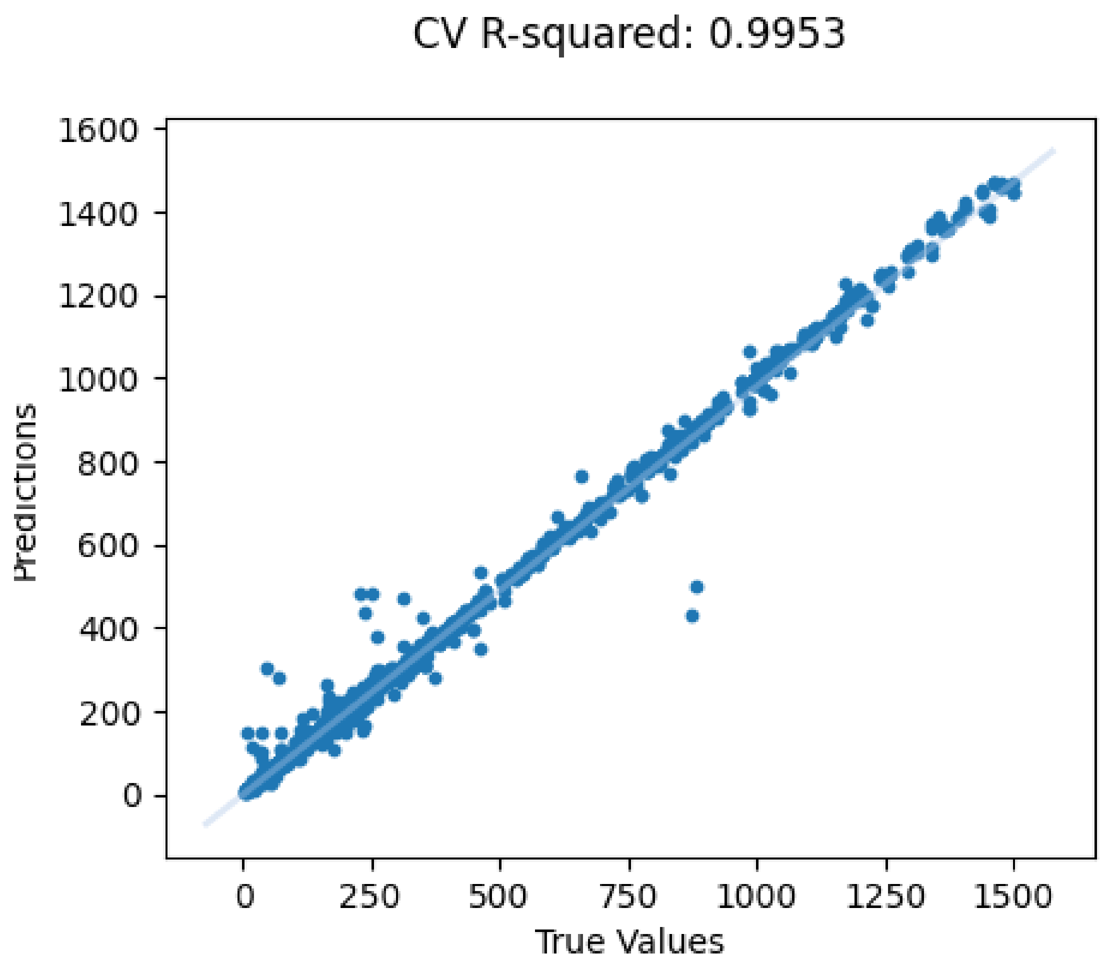
Supervised Learning

```
XGBr = xg.XGBRegressor(learning_rate=0.1, n_estimators=900)
XGBr.fit(X_train_calls, y_train_calls)
XGBr_pred = XGBr.predict(X_test_calls)
```

DecisionTreeRegressor

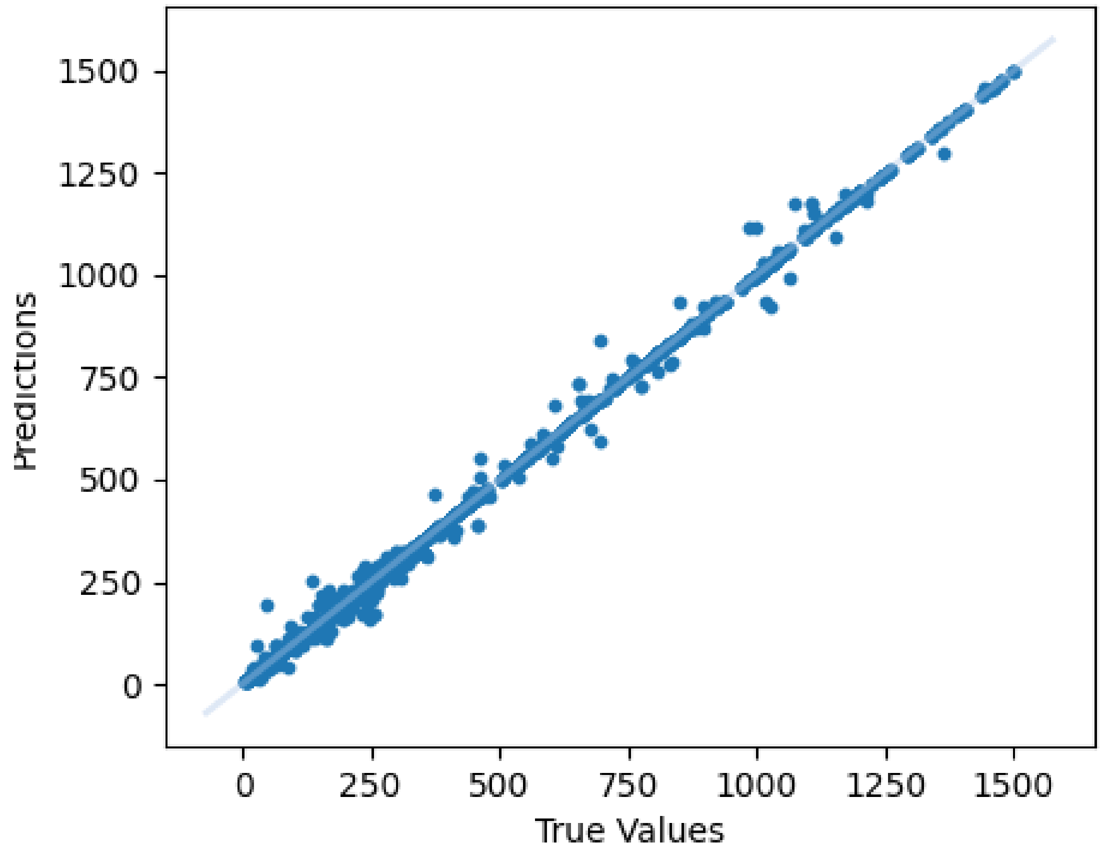
More here

```
DTR_calls = DecisionTreeRegressor()
DTR_calls.fit(X_train_calls, y_train_calls)
DTR1_pred = DTR_calls.predict(X_test_calls)
DTR1_rmse = np.sqrt(mean_squared_error(y_test_calls, DTR1_pred))
```



```
Best Param: {'xgb__colsample_bytree': 1, 'xgb__gamma': 0.01, 'xgb__max_depth': 5, 'xgb__min_child_weight': 3, 'xgb__subsample': 0.6}
```

CV R-squared: 0.9982



```
Best Param: {'dt__max_depth': 16, 'dt__min_samples_leaf': 1}
```

Pipeline

```
def model(pipeline, parameters, X_train, y_train, X, y, figname):
    grid_obj = GridSearchCV(estimator = pipeline, param_grid = parameters, cv = 5, scoring = 'r2', verbose = 0, n_jobs = 1, refit = True)
    grid_obj.fit(X_train, y_train)

    print("Best Param:", grid_obj.best_params_)
    estimator = grid_obj.best_estimator_
    shuffle = KFold(n_splits = 5, shuffle = True, random_state = 0)
    cv_scores = cross_val_score(estimator, X, y.ravel(), cv=shuffle, scoring='r2')

    y_pred = cross_val_predict(estimator, X, y, cv = shuffle)
    plt.figure(figsize = (5,4))
    plt.scatter(y, y_pred, color = palette[0], s=10)
    xmin, xmax = plt.xlim()
    ymin, ymax = plt.ylim()
    plt.plot([xmin, xmax], [ymin, ymax], color = palette[1], lw = 2, alpha = 0.4)
    plt.xlabel("True Values")
    plt.ylabel("Predictions")
    plt.title('CV R-squared: {:.4f}\n'.format(float(cv_scores.mean())), size = 12)
    plt.savefig('./Presentation files/CV_XGBr_puts_plot.png')
    plt.show()
```

Reasons for the XGBr not behaving as expected

Unbalanced data