PREDICIENDO BITCOIN CON MACHINE LEARNING Y DEEP LEARNING

Xavier Roca Pérez

Programación de Redes Neuronales – Machine Learning

Índice

Motivación y descripción del problema

Al ser un apasionado del mundo de las crypto y blockchain, me interesó aprender a como predecir bitcoin. Entonces busque diversos data sets para poder utilizar inteligencia artificial, como machine learning y deep learning para poder predecirlos mejor. Decidí utilizarlo de manera supervisada que es lo que hemos aprendido en el curso y más adelante, me gustaría utilizar reinforcement learning para que vaya aprendiendo solo, pero eso ya sería un extra.

Cuál es el objetivo inicial deseado?

El objetivo deseado inicial es predecir el bitcoin, mirando diferentes maneras de hacerlo, he visto que mucha gente lo intenta predecir día a día, yo voy a intentarlo predecir cada dos horas.

¿De dónde has extraído los datos?

Los datos los he extraído de la api de binance:

```
'https://api.binance.com/'
```

He cogido los datos desde 2017 hasta el día 05-12-2023 de bitcoin, mas adelante me gustaría coger más datos como comparativas de diferentes crypto del top 10 con bitcoin para añadirlo al data set para que haya mas datos.

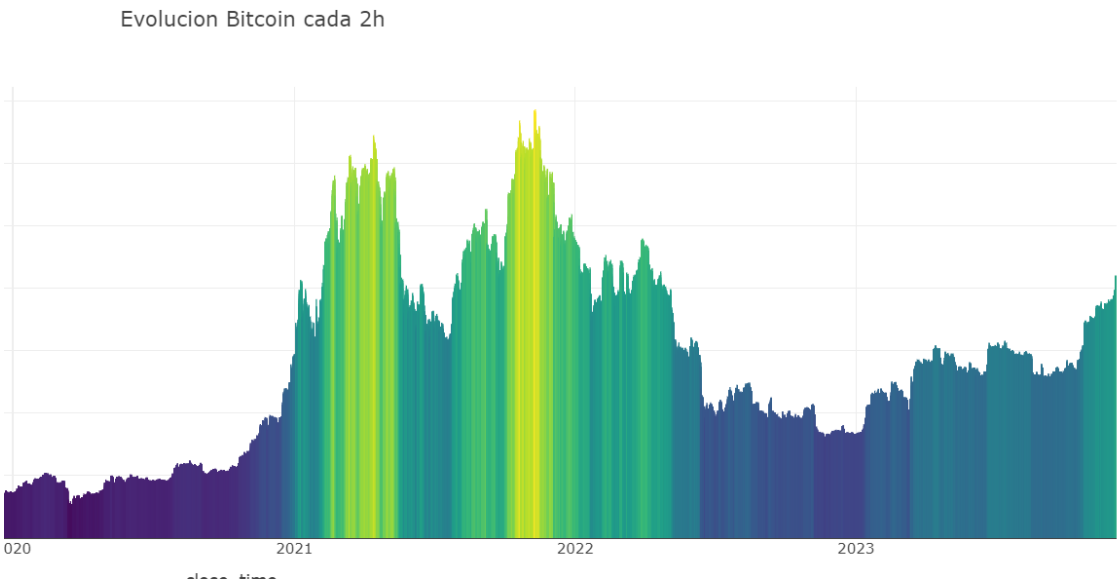
¿Cómo se han limpiado y preparado?

Primero he cogido los datos de la api, después he añadido nombre a las columnas y he pasado la hora de numérica a hora europea en Madrid con el dia que representa. Por ultimo he añadido una columna target que sería lo que tendríamos que predecir que es el precio de close una hacia arriba.

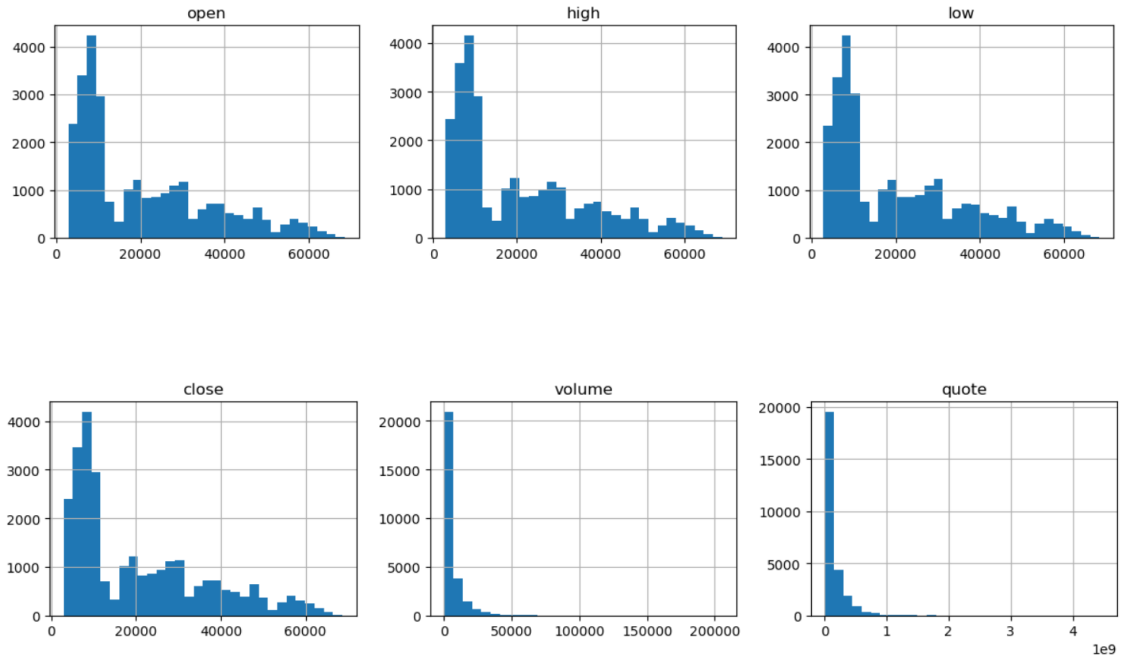
open_time	open	high	low	close	volume
2023-12-05 05:00:00+01:00	41829.98000000	41918.00000000	41724.75000000	41911.09000000	1670.53622000
2023-12-05 07:00:00+01:00	41911.10000000	41919.08000000	41414.00000000	41528.00000000	4831.57684000
2023-12-05 09:00:00+01:00	41528.00000000	41728.00000000	41420.00000000	41705.54000000	2719.06619000
2023-12-05 11:00:00+01:00	41705.55000000	41758.77000000	41565.13000000	41726.09000000	2563.16304000
2023-12-05 13:00:00+01:00	41726.09000000	41864.11000000	41694.00000000	41786.10000000	1258.65010000

close_time	quote	trades	takers_buy_base	takers_buy_quote	ignore	target
2023-12-05 06:59:59.999000+01:00	69863035.35224590	92522	877.45543000	36695932.50422930	0	41528.00000000
2023-12-05 08:59:59.999000+01:00	200961967.19007840	164668	1921.19232000	79943375.99872620	0	41705.54000000
2023-12-05 10:59:59.999000+01:00	113102276.96793730	124334	1342.83865000	55860335.67331080	0	41726.09000000
2023-12-05 12:59:59.999000+01:00	106830010.77133510	98961	1214.70509000	50631074.56260810	0	41786.10000000
2023-12-05 14:59:59.999000+01:00	52598994.99390740	45968	689.62354000	28819660.77755040	0	None

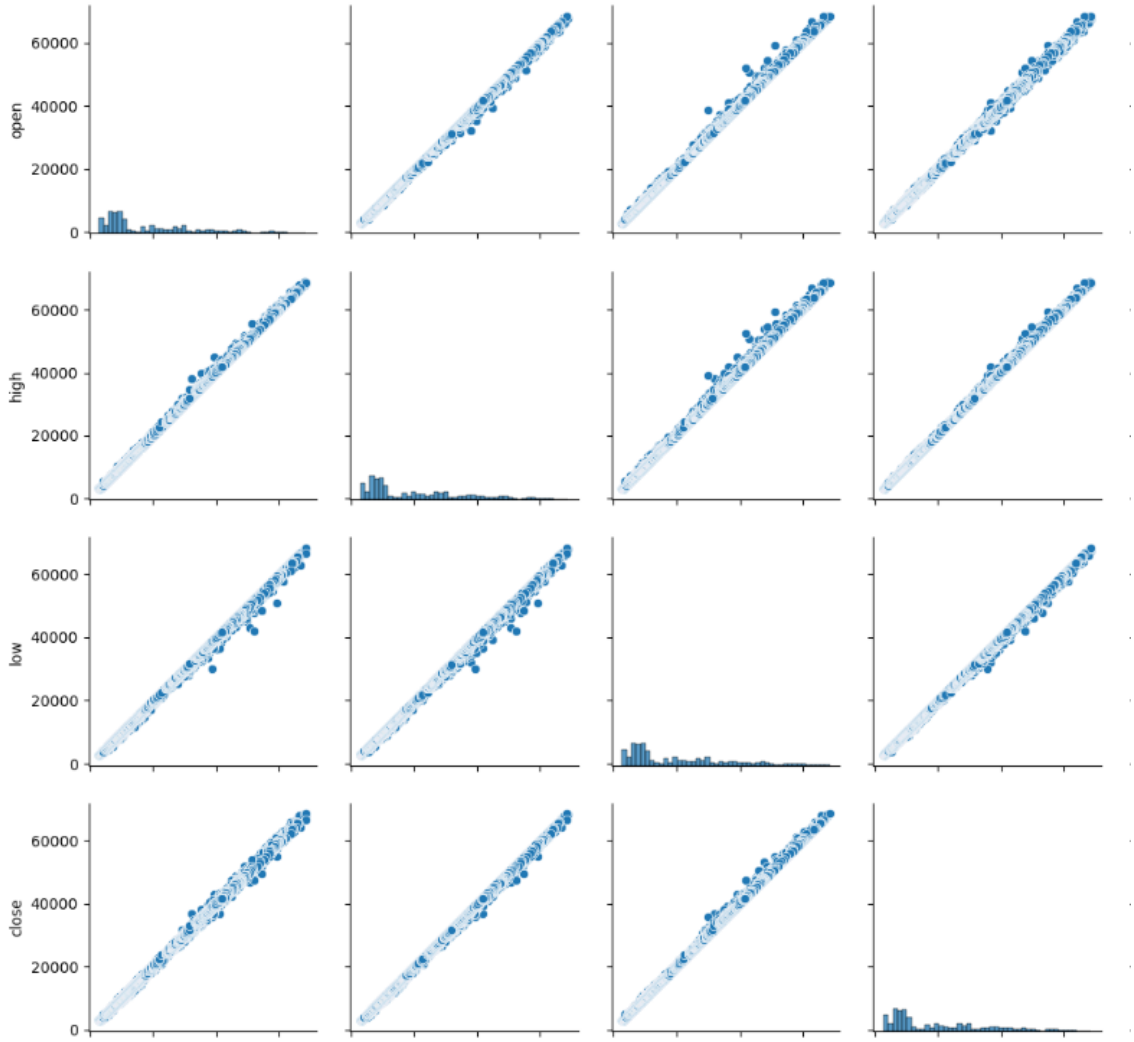
Incluye algunos gráficos relevantes de la exploración de los datos



Numerical features distribution:



Pair plot:



¿Has aplicado “feature engineering”? En caso afirmativo, justifica las decisiones

He hecho una predicción sin nuevos atributos y otro con nuevos atributos que creo que pueden ser interesantes.

price_range	fib_61.8	fib_50	halving	day_of_week	buy_sell
67.37	4287.05534	4295.005	0	3	1.0
62.58	4311.31556	4318.700	0	3	1.0
112.46	4376.27972	4389.550	0	3	1.0
85.58	4432.50156	4442.600	0	3	0.0
48.00	4429.33600	4435.000	0	3	1.0

¿Qué algoritmos de ML has utilizado? ¿Por qué?

He utilizado varios algoritmos de ML, al nosotros querer predecir un numero numérico, he utilizado los algoritmos regressor para ello.

- Linear Regressor
- K-Neighbours
- Decision Tree
- Random Forest
- Gradiant Boosting Decision Tree
- SVR
- Elastic Net
- RNN
    - Simple
    - LSTM
    - Gru

He ido probando varios algoritmos a ver cual daba mejor, las redes neuronales recurrentes no me han funcionado demasiado bien, ya que había un momento que no bajaba nada y no se quedaba en una buena cifra. De los otros algoritmos el que mejor resultado fue Linear Regressor y de ahí pensé que Elastic Net podría mejorarlo, pero al final el Linear Regressor es el que mejor resultado me ha dado.

¿Cómo has realizado el entrenamiento (se ha aplicado algún tipo de cross-validation, hyperparameter tuning, etc.)?

He utilizado un un cross-validation custom y en el hyperparameter tunning he utilizado el randomsearch.

Custom_cv_ts:

```
def custom_cv_ts(df, models, X, y, cv):
    """
    Perform time series cross-validation with a given model, X, y, and cv object.
    """
    # Create empty train and test MAE lists
    train_maes = []
    val_maes = []
    train_mse = []
    val_mse= []
    train_medae = []
    val_medae = []
    best_pa = []
    val_bacc = []
    val_rec = []
    val_prec = []


    # Cross-validation
    for i, (train_index, val_index) in enumerate(cv.split(X)):
        # Train and val data in this split
        X_train, y_train = X.iloc[train_index], y.iloc[train_index]
        X_val, y_val = X.iloc[val_index], y.iloc[val_index]

        # Nested train and validation data from previous train data
        X_train2, X_val2, y_train2, y_val2 = train_test_split(X_train, y_train, test_size=0.1,
                                                              random_state=42, shuffle=False)

        # Calculate validation metrics after training each model
        metrics = []
        for mod in models:
            mod.fit(X_train2, y_train2)
            y_pred_val2 = mod.predict(X_val2)
            metrics.append( mean_absolute_error(y_val2, y_pred_val2) )

        # Select the best model based on validation MAE and retrain it with the whole training data
        best_model_idx = np.argmin(metrics)
        best_model = models[best_model_idx]
        best_params = None
        best_model_str = type(best_model)
        if isinstance(best_model, RandomizedSearchCV) or \
           isinstance(best_model, GridSearchCV):
            best_params = best_model.best_params_
            if isinstance(best_model.best_estimator_, Pipeline):
                # Assuming the model is the 2nd element of the pipeline...
                best_model_str = type(best_model.best_estimator_[1])
            else:
                best_model_str = type(best_model.best_estimator_)
            print('Best validation model:', best_model_idx, f'({best_model_str})')
        if best_params is not None:
            print('Best params:', best_params)
        best_model.fit(X_train, y_train)
```

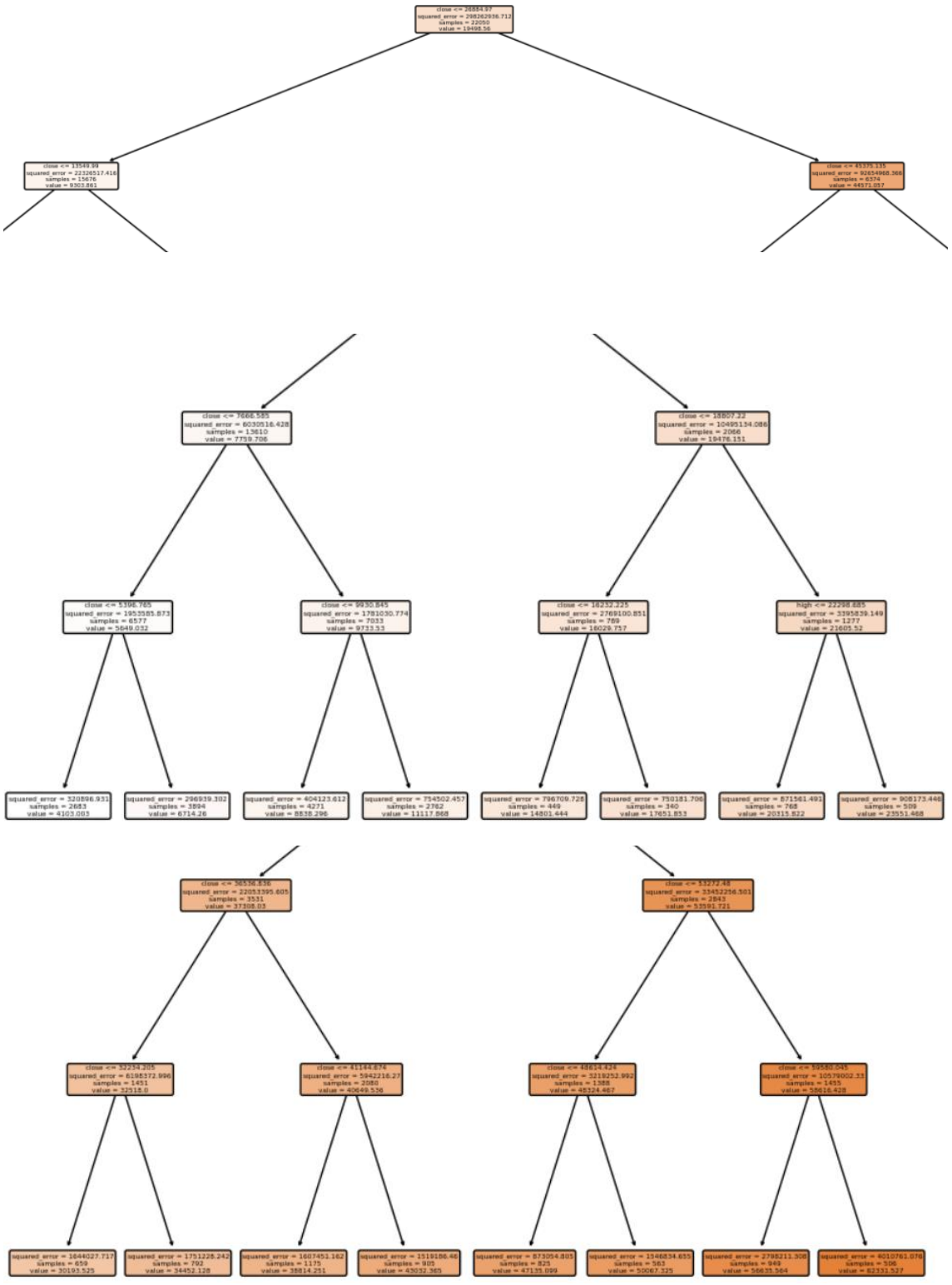
```
    y_diff = y_val2 - X_val2.close
    y_diff
    y_real_bin = pd.Series(np.zeros(len(y_val2)))
    y_real_bin.iloc[np.where(y_diff > 0)[0]] = 1

    #y_pred = lr.predict(X2_test)
    y_diff = y_pred_val2 - X_val2.close
    y_diff
    y_pred_bin = pd.Series(np.zeros(len(y_val2)))
    y_pred_bin.iloc[np.where(y_diff > 0)[0]] = 1

    # Predict and compute MAEs of the train and val sets
    y_pred_train = best_model.predict(X_train)
    y_pred_val = best_model.predict(X_val)
    train_maes.append( mean_absolute_error(y_train, y_pred_train) )
    val_maes.append( mean_absolute_error(y_val, y_pred_val) )
    train_mse.append( mean_squared_error(y_train, y_pred_train) )
    val_mse.append( mean_squared_error(y_val, y_pred_val) )
    train_medae.append( median_absolute_error(y_train, y_pred_train) )
    val_medae.append( median_absolute_error(y_val, y_pred_val) )
    val_bacc.append(balanced_accuracy_score(y_real_bin, y_pred_bin))
    val_prec.append(precision_score(y_real_bin, y_pred_bin))
    val_rec.append(recall_score(y_real_bin, y_pred_bin))

    if isinstance(best_model, RandomizedSearchCV):
        best_pa.append(best_model.best_params_)

    # Show info
    print(1)
    dates = df.loc[X.index, 'open_time']
    dates2 = df.loc[X.index, 'close_time']
    print('Best validation model:', best_model_idx)
    print('Train dates:', dates.iloc[train_index].min().date(), '-', dates.iloc[train_index].max().date())
    print('Train MAE:', round(train_maes[-1], 2))
    print('Train MSE:', round(train_mse[-1], 2))
    print('Train MedAE:', round(train_medae[-1], 2))

    print('Validation dates:', dates.iloc[val_index].min().date(), '-', dates.iloc[val_index].max().date())
    print('Validation MAE:', round(val_maes[-1], 2))
    print('Validation MSE:', round(val_mse[-1], 2))
    print('Validation MedAE:', round(val_medae[-1], 2))
    print('Validation BACC:', val_bacc[-1])
    print('Validation PREC:', val_prec[-1])
    print('Validation REC:', val_rec[-1])

    print()
    if isinstance(best_model, RandomizedSearchCV):
        print("Best Parameters: ",best_model.best_params_)

return np.array(train_maes), np.array(val_maes), np.array(train_mse), np.array(val_mse), np.array(train_medae), np.array(val_medae), np.array(best_pa), np.array(val_bacc), np.array(val_prec), np.array(val_rec),
```

RandomSearch:

```
# Create TimeSeriesSplit object
tscv = TimeSeriesSplit(n_splits=15, test_size=3)

param_dist_gb = {
    "max_depth": [3, 5, 10, 20, 50],
    "min_samples_split": np.arange(2, 30),
    "min_samples_leaf": np.arange(1, 11),
    "criterion": ["friedman_mse", "squared_error"],
    "n_estimators": np.arange(2,200),
    "max_features": ["sqrt", "log2", 0.2, 0.4, 0.6, 0.8],
    "learning_rate": [ 0.08, 0.085, 0.09, 0.099]
}

gb = GradientBoostingRegressor()

# Models to try in each cross-validation split
models = [
    RandomizedSearchCV(gb, param_distributions=param_dist_gb,
                       return_train_score=True,
                       cv=tscv, verbose=1, n_iter=50,
                       n_jobs=-1)
]

train_maes, valid_maes, train_mse, valid_mse, train_medae, valid_medae, best_pa = custom_cv_ts(df, models, X_train, y_train, cv=tscv)
```

¿Qué resultados has obtenido? Incluye algunos gráficos representativas

Decisión tree grafic:



Best models metrics:

```
Mean train MAE: 154.88 $
Mean validation MAE: 125.45 $

Mean train MSE: 92891.17 $
Mean validation MSE: 25939.21 $

Mean train MedAE: 54.64 $
Mean validation MedAE: 111.11 $

Mean validation Bacc: 0.5 %

Mean validation Prec: 0.49 %

Mean validation Rec: 0.48 %
```

Conclusiones

No entendía porque me aba el recall tan bajo, así que lo probé de otra manera

Best models:

```
Mean train MAE: 143.19 $
Mean test MAE: 151.65 $

Mean train MSE: 79767.24 $
Mean test MSE: 55247.76 $

Mean train MedAE: 54.39 $
Mean test MedAE: 107.48 $

Mean test Bacc: 0.51 %

Mean test Prec: 0.57 %

Mean test Rec: 0.1 %

[]
```

Linear Regression: fue el modelo que mejor me fue

```
Test Bacc: 0.5463318884636483
Test Prec: 0.5653043269615292
Test Rec: 0.49862433862433864
```

¿Has logrado el objetivo que te habías propuesto? En caso negativo, ¿qué ha fallado?

He conseguido a medias el objetivo que tenía planteado, ya que he podido ver cómo hacerlo, las métricas balance accuracy y precisión no están mal pero el recall haciendo los tests es muy bajo, entonces hice otras pruebas, y vi que el recall ya me subía al 50%, entonces creo que es una buena métrica, aunque para poder invertir con este modelo, querría que fuera más del 60% o 70% de recall, aun así podría hacer un backtesting para probarlo y ver resultados. Otra cosa que me he dado cuenta es que lo que obtengo en mi mejor modelo con la mejor métrica no mejora de mucho lo obtenido con la baseline, entonces nos damos cuenta que no es un gran modelo.

Describe brevemente cuáles podrían ser los siguientes pasos a realizar para mejorar el proyecto

Los siguientes pasos a solucionar son, intentar descubrir una red neuronal que pueda funcionar y ampliar el data set con más datos para que le sea más fácil predecir el precio cada dos horas, hacer backtesting para probar si con el modelo se puede ganar dinero.