STOCK PREDICTION

Yan Jun Chen Ye

ÍNDICE

INTRODUCCIÓN

Nuestro objetivo es calcular el precio ajustado de cierre de diferentes empresas

Empresas las cuales vamos a predecir:

- Apple
- Google
- Inditex
- Johnson & Johnson
- JPMorgan
- Tesla

Herramientas usadas:

Python, Pandas, Statsmodels, XGBoost, Scikit-learn, Matplotlib.

DATOS USADOS

Obtención de datos

```
api = APIClient(api_key='68a888ecbf2185.97717291')

TICKERS = ["AAPL.US", "GOOGL.US", "TSLA.US", "JNJ.US", "JPM.US", "ITX.MC"]

os.makedirs("../data/raw", exist_ok=True)

for i in TICKERS:
    df = pd.DataFrame(api.get_eod_historical_stock_market_data(symbol= i, period='d', from_date='2010-01-01'))
    df.to_csv(f'../data/raw/{i}.csv', index=False)
```

import os

from eodhd import APIClient

DATOS USADOS

Feature Engineering

	date	open	high	low	close	adjusted_close	volume
0	2010-01-04	213.4300	214.4996	212.3800	214.0096	6.4246	493729600
1	2010-01-05	214.6004	215.5888	213.2508	214.3792	6.4357	601904800
2	2010-01-06	214.3792	215.2304	210.7504	210.9688	6.3333	552160000
3	2010-01-07	211.7500	211.9992	209.0508	210.5796	6.3216	477131200
4	2010-01-08	210.2996	211.9992	209.0592	211.9796	6.3637	447610800
...	...	...	...	...	...	...	...
3934	2025-08-25	226.4800	229.3000	226.2300	227.1600	227.1600	30983100
3935	2025-08-26	226.8700	229.4900	224.6900	229.3100	229.3100	54575100
3936	2025-08-27	228.6100	230.9000	228.2600	230.4900	230.4900	31259500
3937	2025-08-28	230.8200	233.4100	229.3400	232.5600	232.5600	38074700
3938	2025-08-29	232.5100	233.3800	231.3700	232.1400	232.1400	39389400

date	open	adjusted_close	volume	close_diff_1	dayofweek	quarter	month	year	dayofyear	dayofmonth	weekofyear	adjusted_close(-1)	SMA	EMA	MACD	rsi_14	H_L_diff	Bands_diff	target
2025-08-25	226.48	227.16	30983100	-0.60	0	3	8	2025	237	25	35	227.76	228.513892	226.995376	5.836679	1.535379	3.07	11.524136	229.31
2025-08-26	226.87	229.31	54575100	2.15	1	3	8	2025	238	26	35	227.16	229.246923	227.458301	5.938659	1.715462	4.80	11.522352	230.49
2025-08-27	228.61	230.49	31259500	1.18	2	3	8	2025	239	27	35	229.31	229.354615	228.064641	6.060884	1.821901	2.64	10.309129	232.56
2025-08-28	230.82	232.56	38074700	2.07	3	3	8	2025	240	28	35	230.49	229.768462	228.963713	6.238277	2.022982	4.07	10.174094	232.14
2025-08-29	232.51	232.14	39389400	-0.42	4	3	8	2025	241	29	35	232.56	229.960000	229.598970	6.358737	1.937838	2.01	10.431709	NaN

ENTRENAMIENTO Y MODELO

```python
def train_test_split(df, test_size=0.2):
    data = df.values

    feature_scaler.fit(data[:, :-1])
    target_scaler.fit(data[:, -1:])
    scaled_data = feature_scaler.transform(data[:, :-1])
    scaled_target = target_scaler.transform(data[:, -1:])
    data_scaled = np.concatenate((scaled_data, scaled_target), axis=1)

    n = int(len(data_scaled) * (1 - test_size))
    return data_scaled[:n], data_scaled[n:]
```

ENTRENAMIENTO Y MODELO

```python
def xgb_prediction(train, value):
    train = np.array(train)
    X, Y = train[:, :-1], train[:, -1]
    global model
    model = XGBRegressor(objective='reg:squarederror', tree_method = 'hist', device = 'cpu', n_jobs = -1 ,n_estimators=1000)

    model.fit(X, Y)
    val = np.array(value).reshape(1, -1)
    prediction = model.predict(val)
    return prediction[0]
```

ENTRENAMIENTO Y MODELO

```python
def walk_forward_validation(data, percentage=0.2):
    train, test = train_test_split(data, percentage)
    predictions = []
    history = [x for x in train]

    for i in range(len(test)):
        test_X, test_Y = test[i, :-1], test[i, -1]
        pred = xgb_prediction(history, test_X)
        predictions.append(pred)
        history.append(test[i])

    Y_test = target_scaler.inverse_transform(test[:, -1:].reshape(1, -1))
    Y_pred = target_scaler.inverse_transform(np.array(predictions).reshape(1, -1))
    test_rmse = metrics.root_mean_squared_error(Y_test, Y_pred)

    return test_rmse, Y_test, Y_pred
```

ENTRENAMIENTO Y MODELO

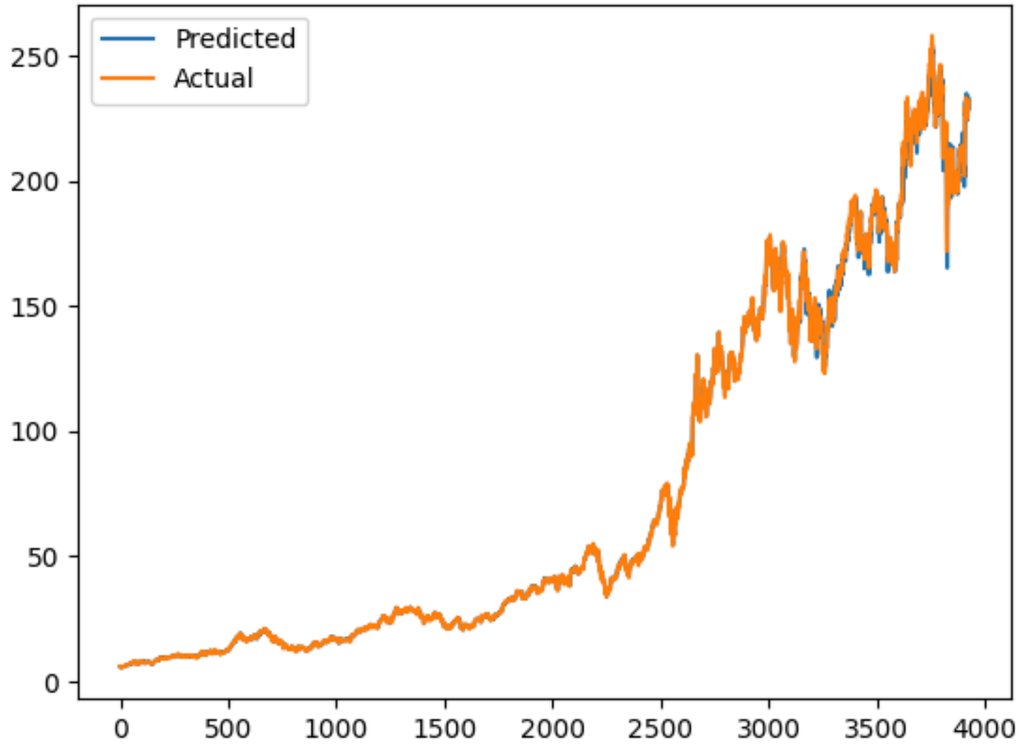
```
test_rmse_aapl, Y_test_aapl, predictions_aapl = walk_forward_validation(df_aapl, 0.2)

aapl_model = model
aapl_model.save_model("../models/aapl_model.json")
```

Así con todas las empresas.

EVALUACIÓN Y RESULTADOS

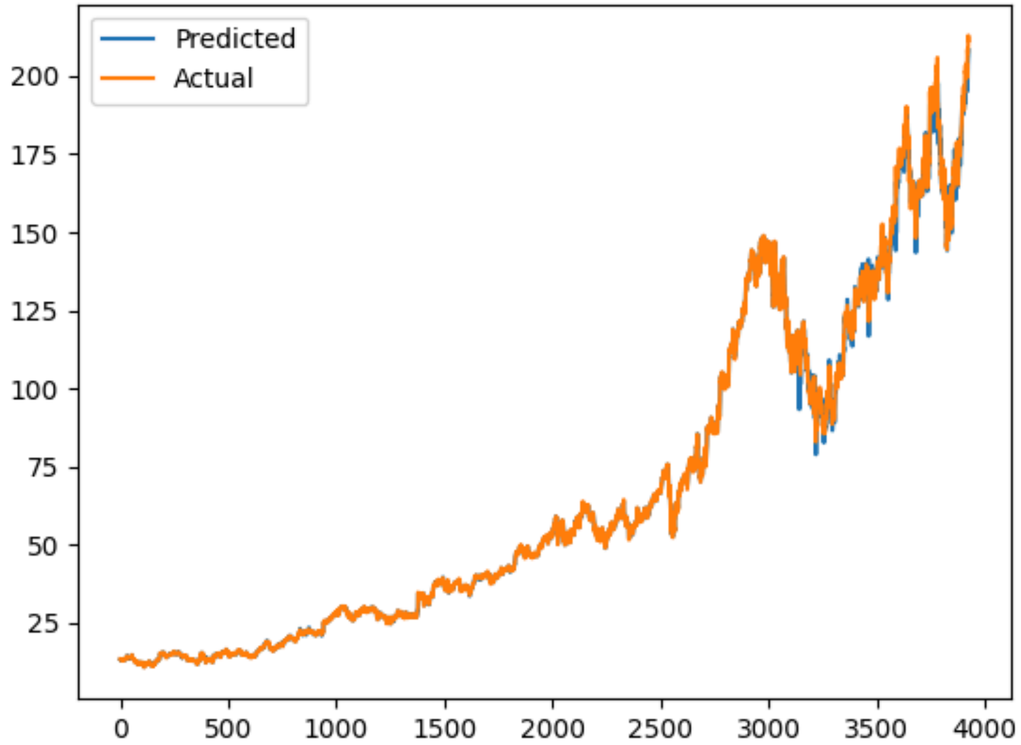
APPLE




```
Training RMSE: 0.29082564822099716
Testing RMSE: 3.1195750148972436
```

EVALUACIÓN Y RESULTADOS

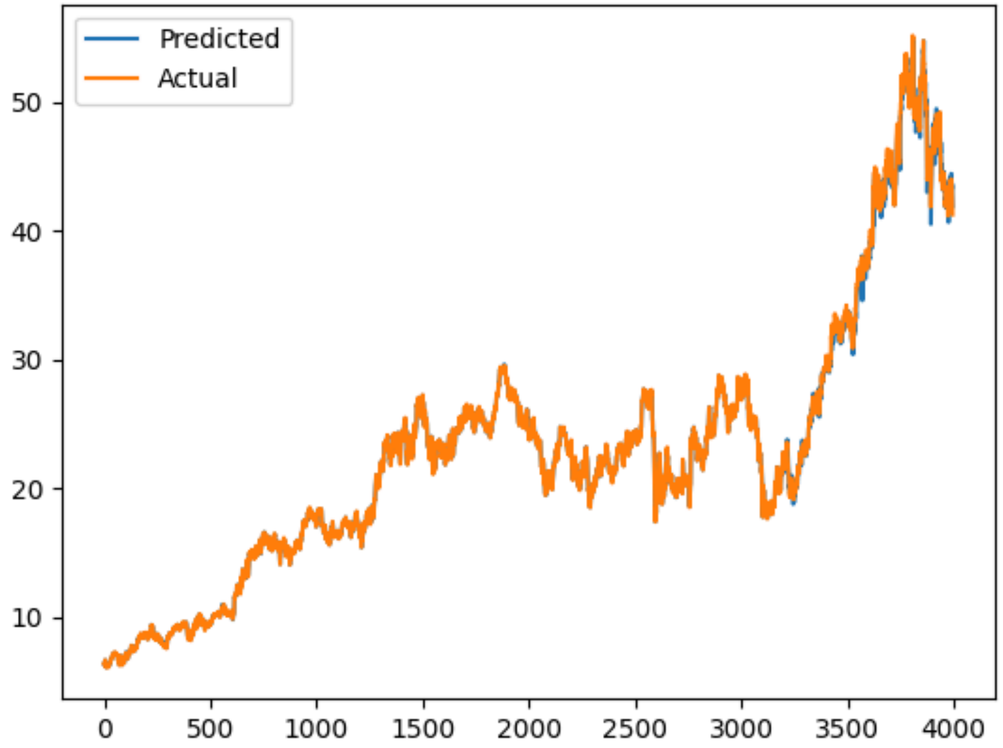
GOOGLE




```
Training RMSE: 0.2782262786484691
Testing RMSE: 2.7474460135101513
```

EVALUACIÓN Y RESULTADOS

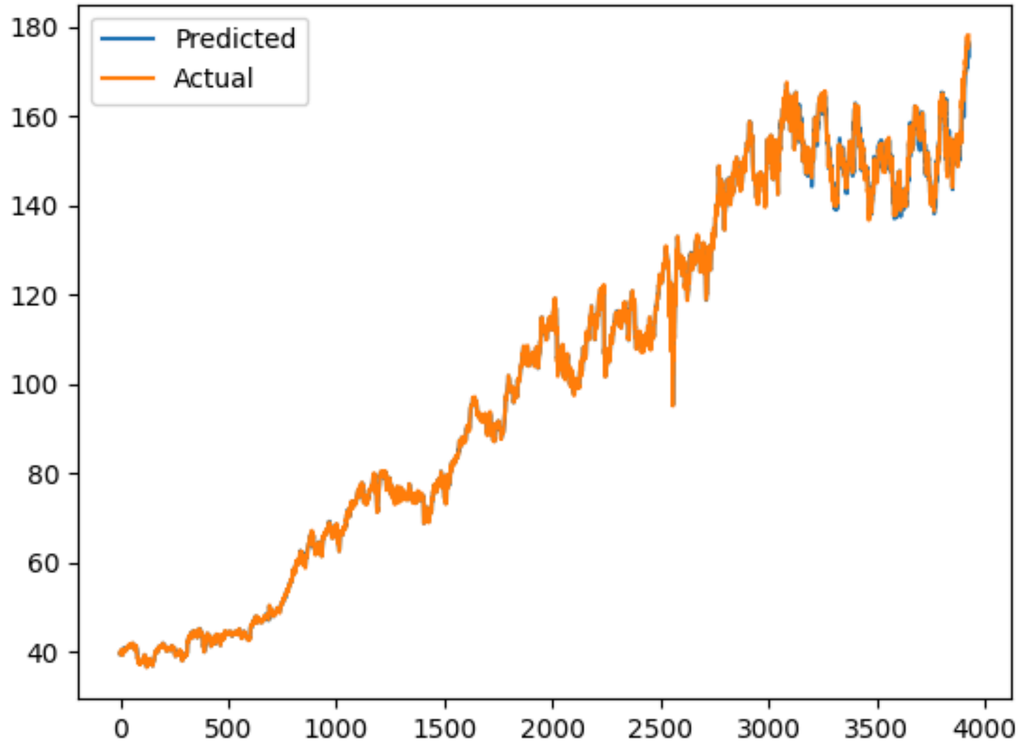
INDITEX




```
Training RMSE: 0.06735533661859477
Testing RMSE: 0.5369720493168647
```

EVALUACIÓN Y RESULTADOS

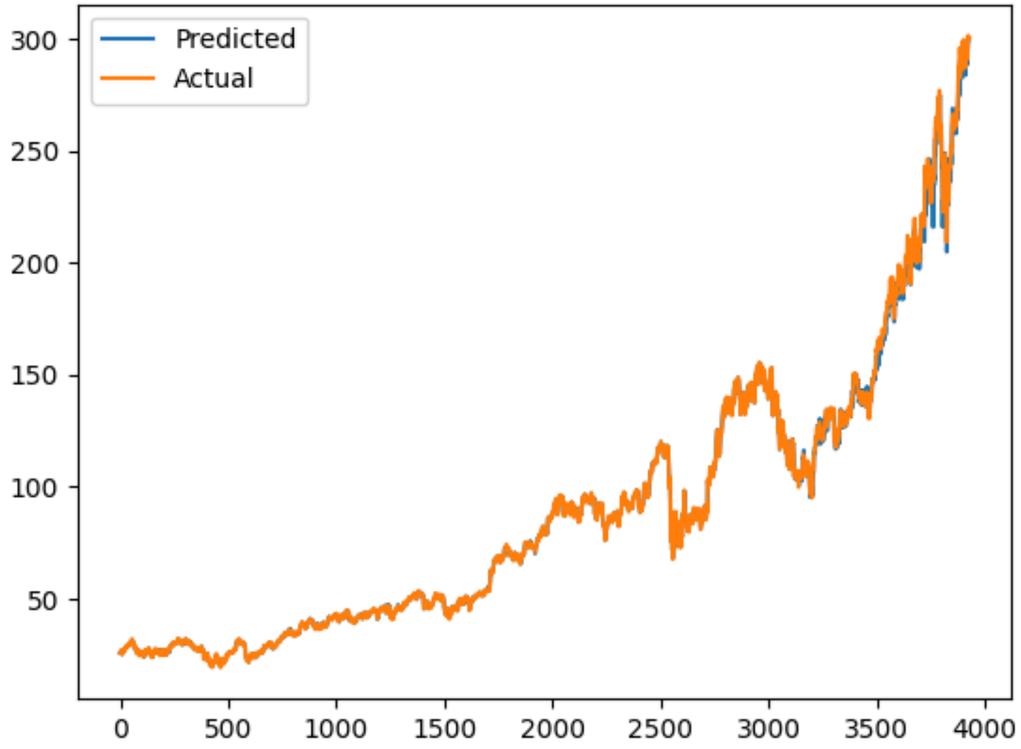
JOHNSON & JOHNSON




```
Training RMSE: 0.21702156486078847
Testing RMSE: 1.4294687426602308
```

EVALUACIÓN Y RESULTADOS

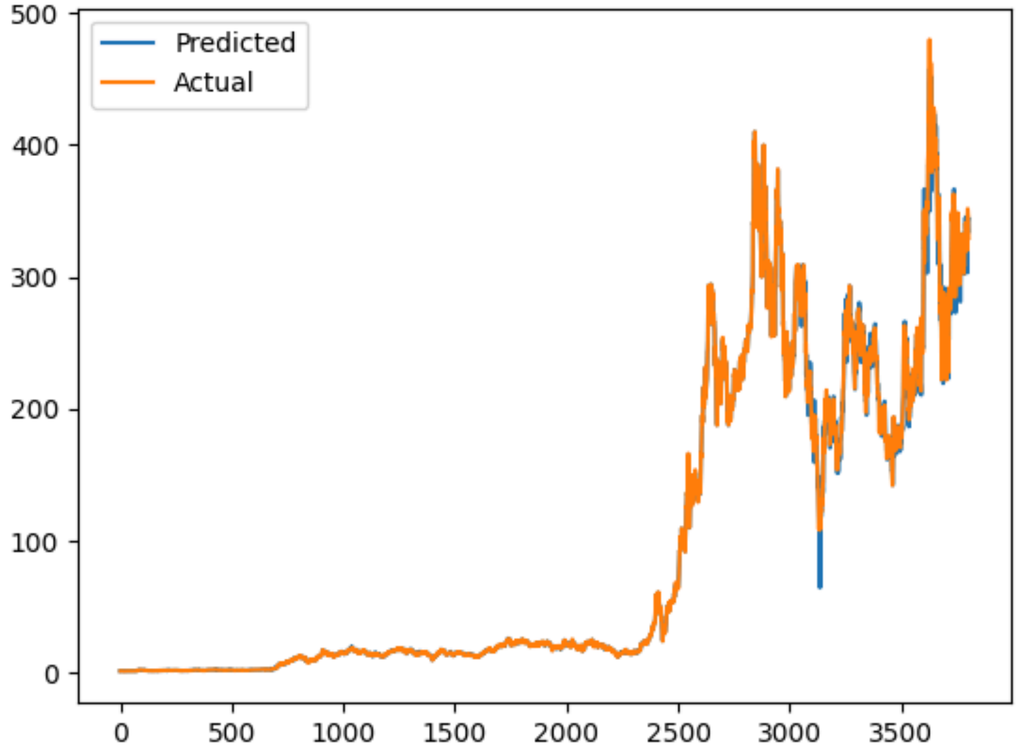
JPMORGAN




```
Training RMSE: 0.37707112929802156
Testing RMSE: 2.75604320272069
```

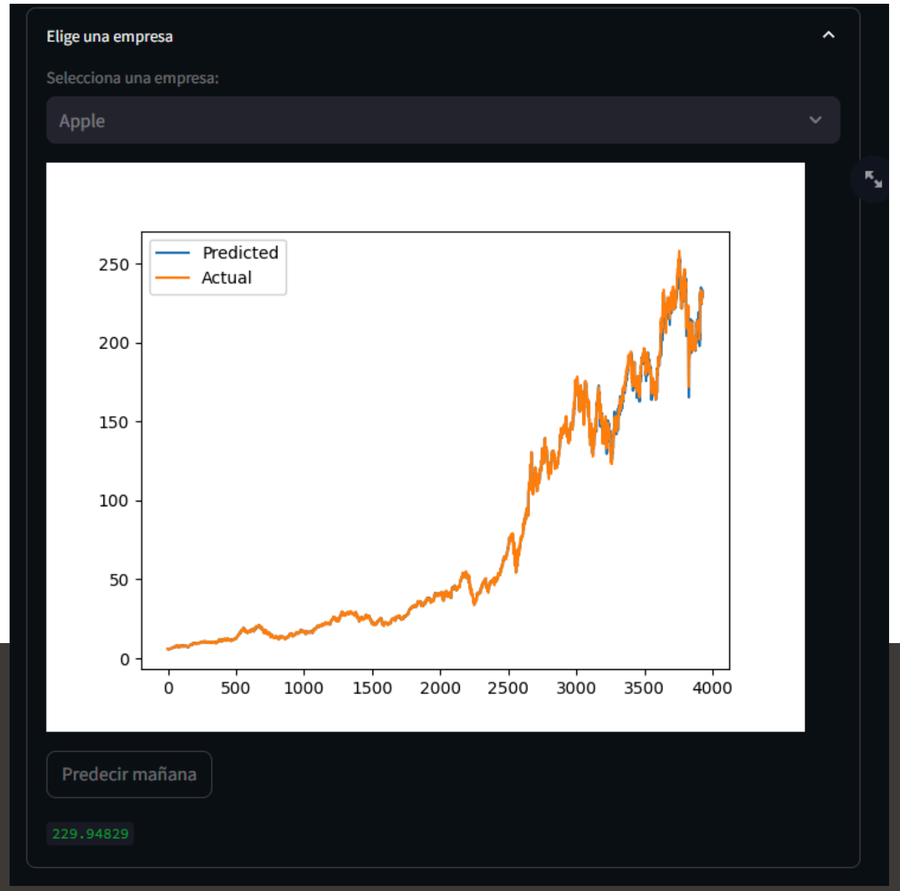
EVALUACIÓN Y RESULTADOS

TESLA




```
Training RMSE: 0.44693448978212924
Testing RMSE: 8.040566819283967
```

PREDICCIÓN FINAL



CONCLUSIÓN

- El modelo captura patrones relevantes de series temporales financieras.
- Limitaciones:
    - No considera noticias, eventos externos ni correlaciones globales.
    - Basado solo en datos históricos y técnicos.
- Futuras mejoras:
    - Inclusión de NLP en noticias financieras.
    - Uso de LSTM/Transformers para series temporales.
    - Optimización de hiperparámetros de XGBoost. (GridSearch)

¡¡¡ADVERTENCIA!!!

Este proyecto y los modelos presentados han sido desarrollados únicamente con fines educativos y de investigación académica.

No deben interpretarse ni utilizarse como recomendaciones financieras, de inversión o de trading.

El uso de estos resultados para la toma de decisiones económicas reales puede conllevar pérdidas financieras.

El autor no se hace responsable de cualquier uso indebido de la información contenida en este trabajo.

GRACIAS POR SU ATENCIÓN