ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

Μηχανική Μάθηση (7ο εξάμηνο)

Απαλλακτική Εργασία : Πρόβλεψη Μελλοντικών τιμών BTC τύπου Regression

Ημερομηνία: 17/1/2025

Φοιτητές: Παναγιώτης Μώκος (iis22125) Δαδακίδης Γιώργος (iis22127)

Επιβλέπων Καθηγητής: Πρωτοπαπαδάκης Ευτύχιος

Τμήμα Εφαρμοσμένης Πληροφορικής Πληροφοριακά Συστήματα Πανεπιστήμιο Μακεδονίας

Περιεχόμενα

1.Προσδιορισμός Προβλήματος

2.Συλλογή Δεδομένων

3.Εξερεύνηση και Ανάλυση Δεδομένων

4.Προ Επεξεργασία Δεδομένων

5.Επιλογή Μοντέλων - Αρχική τοποθέτηση/Σχεδιασμός Μοντέλων

6.Βελτιστοποίηση Μοντέλων

- Βελτιστοποίηση Linear Regression
- Βελτιστοποίηση Logarithmic Fit
- Βελτιστοποίηση Exponential Fit
- Βελτιστοποίηση kNN
- Βελτιστοποίηση Random Forest
- Βελτιστοποίηση SVR
- Βελτιστοποίηση LSTM

7.Γενική Ανάλυση – Επιπλέον Βελτιστοποιήσεις

- Τεχνική Ανάλυση
- Θεμελιώδης Ανάλυση
- Ψυχολογική Ανάλυση
- Τελική Ανάλυση

8.Αξιολογήσεις Μοντέλων

9.Τελικές Προβλέψεις Μοντέλων

10.Σύγκριση Μοντέλων

Link για τους κώδικες σε google colab:

- ***https://colab.research.google.com/drive/14beKTqjXzMiYmLrp0GqDGruyN4khiqbi?usp=sharing***

Σημείωση:

Έχουν Χρησιμοποιηθεί λίγοι (σαν λογική) κώδικες από τα εργαστήρια. Παίρναμε Βοήθεια από οπουδήποτε υπήρχε γενικότερο υλικό και βοήθεια.

Γιατί είναι σημαντικό;

Οι Προβλέψεις Τιμών(Regression) Μπορεί να Ποικίλουν από προβλέψεις μετοχών μέχρι και προβλέψεις μελλοντικών σημαντικών πληροφοριών όπως την ώρα που κάποιος ασθενής χρειάζεται να πάρει ένα χάπι και πολλά άλλα ακόμα..

Τα κίνητρα Στα συγκεκριμένα προβλήματα συνήθως Αφορούν Προσωπικό όφελος τόσο σε χρήματα όσο και στην γενικότερη γνώση της κατεύθυνσης όλης της αγοράς.

Βασιστήκαμε Κυρίως στην Λογική του Rainbow Chart

https://www.blockchaincenter.net/en/bitcoin-rainbow-chart/

1.Προσδιορισμός Προβλήματος

<u>Εισαγωγή</u>

Το έργο αυτό στοχεύει στην πρόβλεψη μελλοντικών τιμών του Bitcoin μέσω επιβλεπόμενων μοντέλων παλινδρόμησης. Χρησιμοποιώντας ιστορικά δεδομένα, Τεχνική, Θεμελιώδη, Ψυχολογική Ανάλυση και τεχνικές μηχανικής μάθησης, επιδιώκεται η ανάπτυξη προβλέψεων που ενδεχομένως ΕΑΝ ΠΡΟΦΑΝΩΣ ΑΝΑΠΤΥΧΘΟΥΝ ΣΤΟ ΜΕΛΛΟΝ μπορούν να υποστηρίξουν επενδυτικές αποφάσεις.

<u>Στόχος</u>

Η ανάπτυξη και αξιολόγηση μοντέλων επιβλεπόμενης μάθησης για την ακριβή πρόβλεψη των τιμών του Bitcoin.

<u>Δεδομένα</u>

Το σύνολο δεδομένων περιλαμβάνει:

- ***Ημερομηνία***
- ***Τιμές (Άνοιγμα, Κλείσιμο, Υψηλό, Χαμηλό)***
- ***Δείκτες***
- ***Μετρικές***

<u>Αναμενόμενα Αποτελέσματα</u>

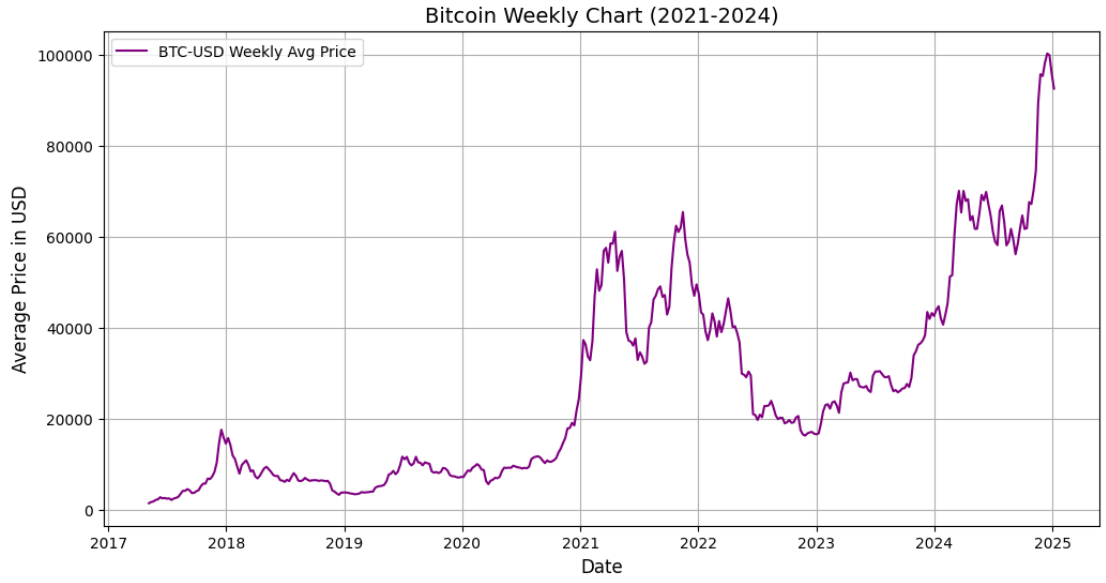
- ***Ακριβή πρόβλεψη της τελευταίας τιμής του Bitcoin από το test dataset(ώστε να είναι εφικτή η σύγκριση με την πραγματική του τιμή).***
- ***Ανάλυση απόδοσης διαφορετικών μοντέλων.***
- ***Εντοπισμός παραγόντων που επηρεάζουν τις τιμές.(Τεχνική , Θεμελιώδη , Ψυχολογική Ανάλυση)***

2.Συλλογή Δεδομένων

Συλλογή & Αρχική Προεπεξεργασία Δεδομένων.
Τα ιστορικά δεδομένα του Bitcoin συλλέχθηκαν από το Yahoo Finance χρησιμοποιώντας την Python βιβλιοθήκη yfinance:

```
btc_data = yf.download("BTC-USD", start="2017-05-01", end="2024-12-31")
```

Entire Dataset.



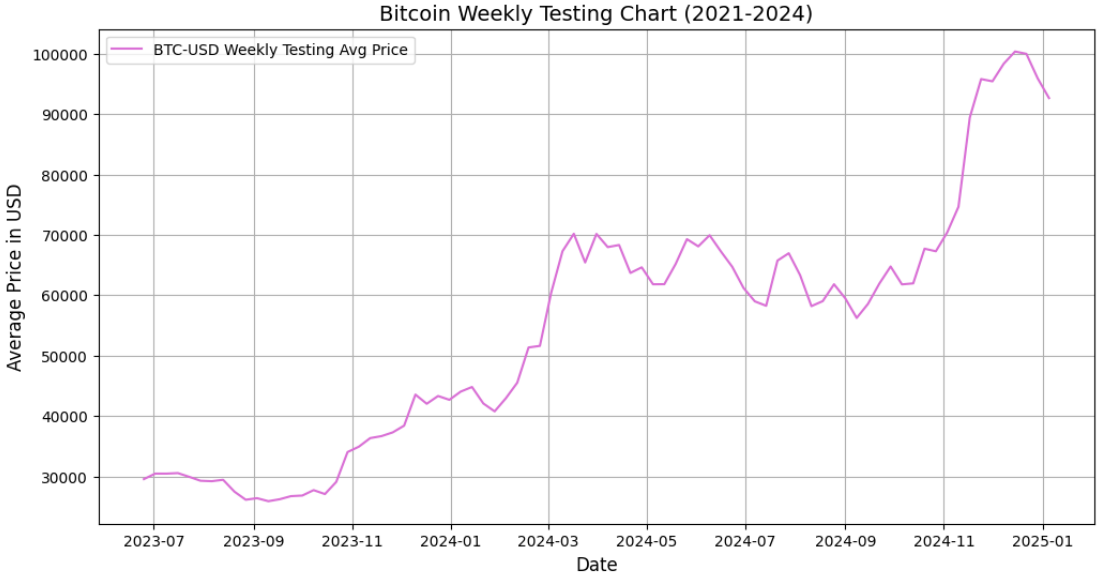
Όπου και στην πορεία χωρίστηκαν σε train & test data:

```
btc_data_train = yf.download("BTC-USD", start="2017-05-01", end="2023-06-19")
btc_data_test = yf.download("BTC-USD", start="2023-06-19", end="2024-12-31")
```

(ΕΠΟΜΕΝΩΣ SPLIT 80-20%)

Train Dataset

Test Dataset

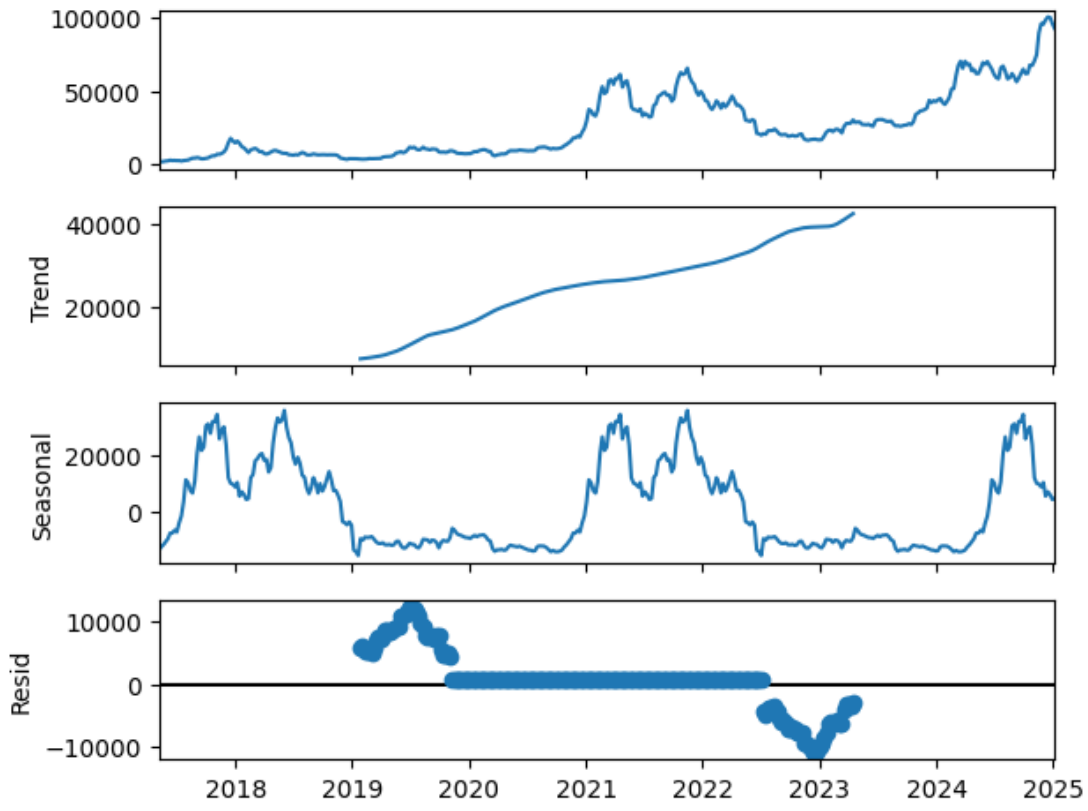


3.Εξερεύνηση και Ανάλυση Δεδομένων

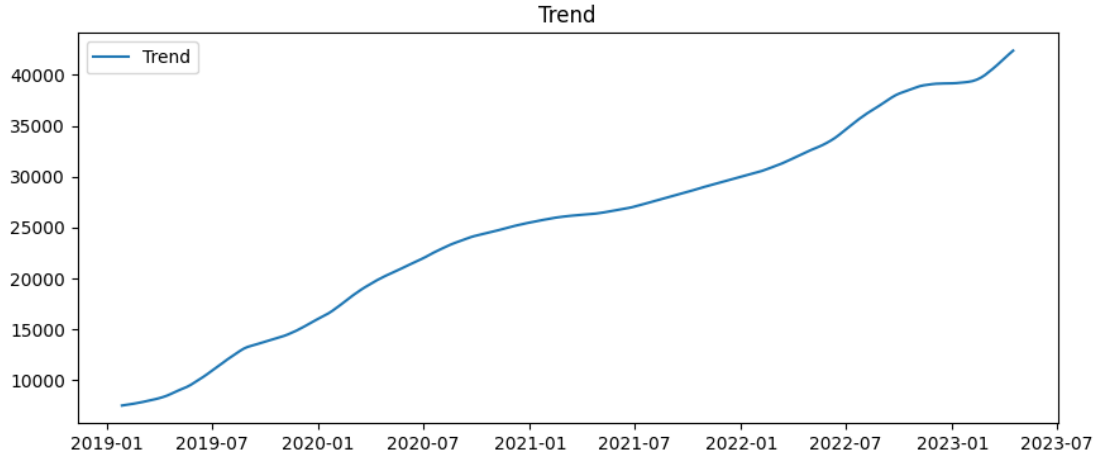
Πραγματοποιείται Ανάλυση χρονοσειρών(Time Series Analysis) για την καλύτερη κατανόηση της συμπεριφοράς των τιμών αλλά και της φύσης και γενικότερης κατεύθυνσης του Bitcoin. Συγκεκριμένα, εξετάστηκαν τα εξής στοιχεία:

- ***Γενική Τάση (Trend): Αναλύθηκε η μακροπρόθεσμη πορεία των τιμών για να εντοπιστούν αυξητικές ή πτωτικές τάσεις.***

```
result = seasonal_decompose(btc_weekly, model='additive', period=180)  # Weekly data with high timeframe period to identify the trend
```



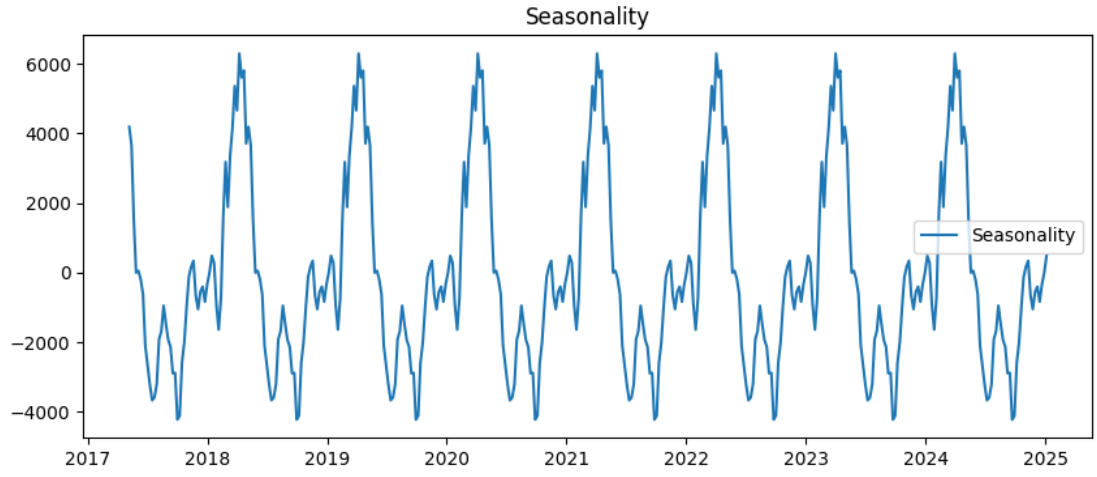
Επομένως Trend :



- ***Εποχικότητα (Seasonality): Εντοπίστηκαν περιοδικά μοτίβα στις τιμές, τα οποία επαναλαμβάνονται σε συγκεκριμένα χρονικά διαστήματα.***

```
result = seasonal_decompose(btc_weekly, model='additive', period=52)  # Weekly data with ~1 year seasonality
```

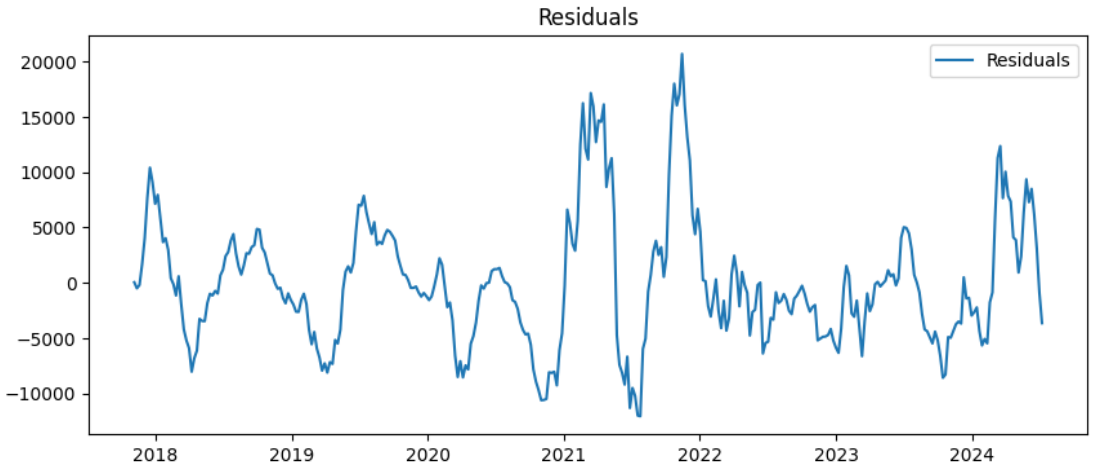
Seasonality:



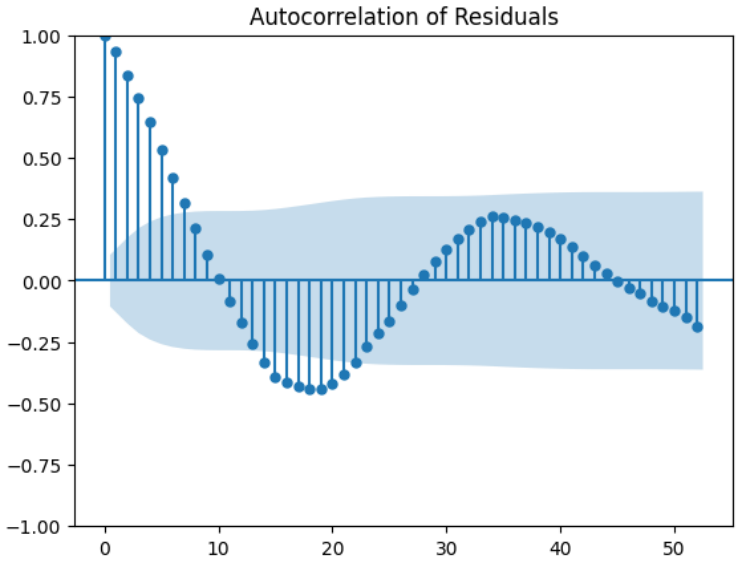
- ***Σφάλματα – Αποκλίνουσες Τιμές (Residuals): Μελετήθηκαν οι τυχαίες διακυμάνσεις που δεν εξηγούνται από τη γενική τάση και την εποχικότητα.***

```
result = seasonal_decompose(btc_weekly, model='additive', period=52)  # Weekly data with ~1 year seasonality
```

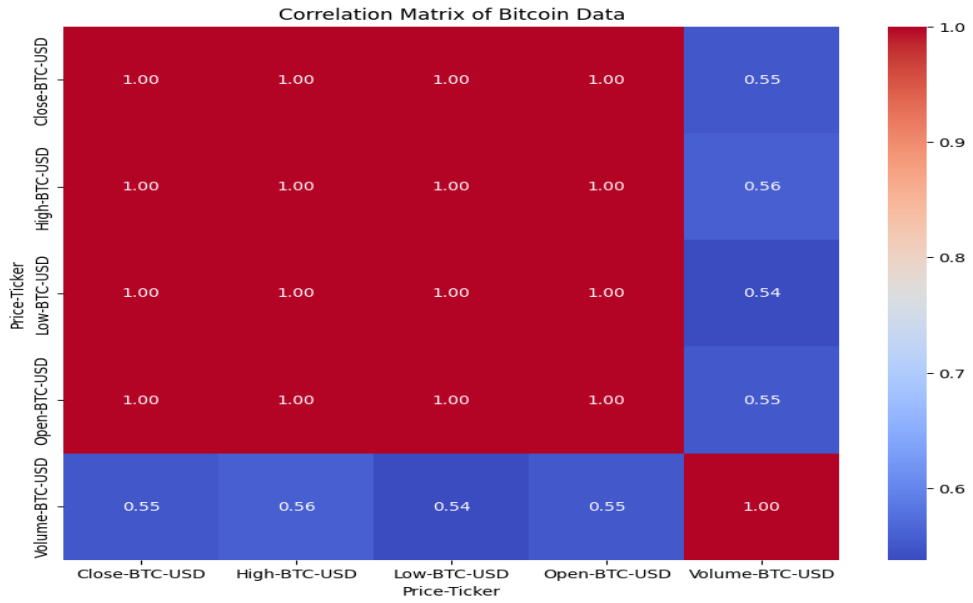
Residuals:

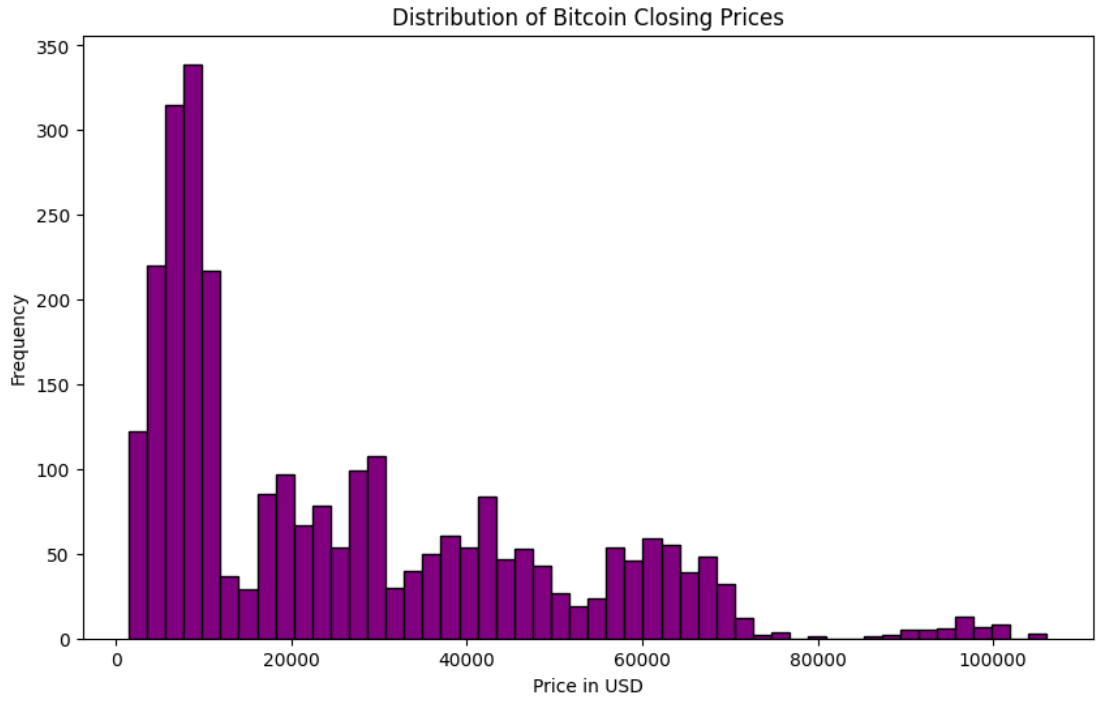


Συσχέτιση μεταξύ Residuals:



- ***Άλλα Visualizations***





4.Προ Επεξεργασία Δεδομένων

Κατά Κύριο λόγο αφού τα δεδομένα τα παίρνουμε μέσω API του yahoofinance δεν έχουν πολλές λάθος τιμές οπότε η προεπεξεργασία δεδομένων είναι λίγο πιο εύκολη.

```
# Check for missing values
print(btc_data.isnull().sum())
print(btc_data_train.isnull().sum())
print(btc_data_test.isnull().sum())

# Fill missing values with forward fill (you can also try interpolation or removal)
btc_data = btc_data.fillna(method='ffill')

# Feature Engineering: Adding technical indicators
# Convert btc_data['Close'] to a pandas Series
btc_data['SMA_30'] = ta.trend.sma_indicator(btc_data['Close'].squeeze(), window=30)  # Simple Moving Average
btc_data['SMA_100'] = ta.trend.sma_indicator(btc_data['Close'].squeeze(), window=100)
btc_data['RSI'] = ta.momentum.rsi(btc_data['Close'].squeeze(), window=14)  # Relative Strength Index
btc_data['MACD'] = ta.trend.macd(btc_data['Close'].squeeze())  # MACD
btc_data['MACD_signal'] = ta.trend.macd_signal(btc_data['Close'].squeeze())  # MACD Signal Line

# Drop rows with missing values that may result from technical indicators
btc_data.dropna(inplace=True)

# Feature selection: We'll use the features for prediction
features = ['Close', 'SMA_30', 'SMA_100', 'RSI', 'MACD', 'MACD_signal'] # This line remains unchanged
```

Επίσης Χρησιμοποιείται η βιβλιοθήκη ta για αρχικό feature engineering μέσω των τεχνικών δεικτών SMA30,SMA100,RSI,MACD.

5.Επιλογή Μοντέλων - Αρχική τοποθέτηση/Σχεδιασμός Μοντέλων

Θα Γίνει Ανάπτυξη και Χρήση των Εξής Μοντέλων και Τεχνικών Μηχανικής Μαάθησης (και γενικότερα μαθηματικών):

- ***Linear Regression***
- ***Logarithmic Fit***
- ***Exponential Fit***
- ***kNN***
- ***SMA50***
- ***Random Forest***
- ***SVR***
- ***LSTM***

Χρησιμοποιούνται τόσο 'απλά' Μοντέλα όσο και Μοντέλα Deep Learning(LSTM) για λόγους Ποικιλίας.

Επίσης από Μετρικές Αξιολόγησης:

- ***RMSE***
- ***MSE***
- ***MAE***
- ***MAPE***
- ***R^2***

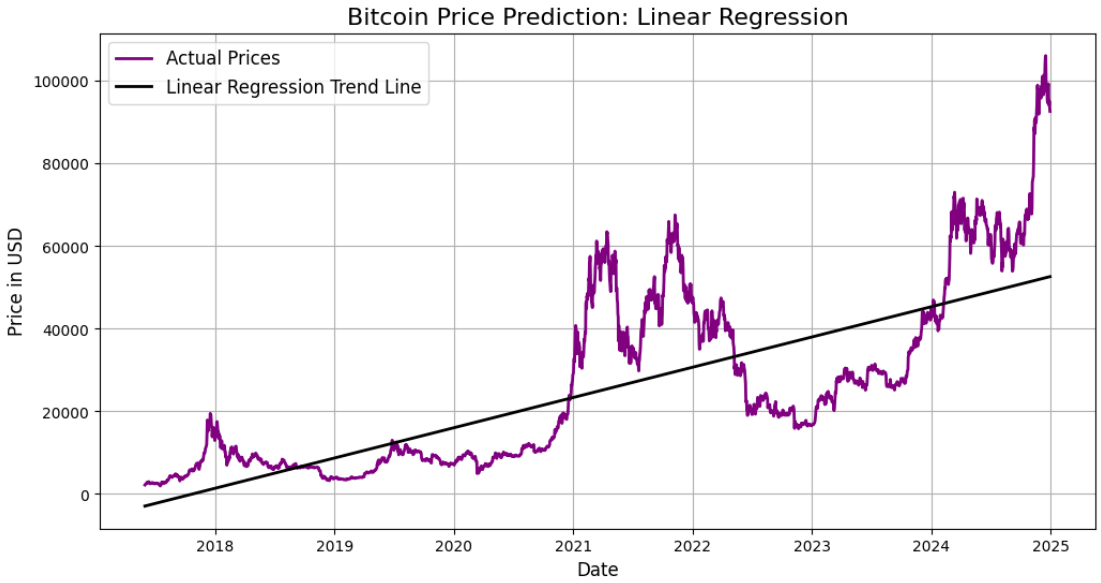
Το αρχικό 'Fit' των Μοντέλων γίνεται με βάση τα features που αναφέραμε προηγουμένως και με βάση κάποιων παραμέτρων(υπερπαραμέτρων) που διαλέξαμε τυχαία.

Όλα τα Μοντέλα ακολουθούν την αντίστοιχη λογική(όμως ρυθμίζονται στις αντίστοιχες ανάγκες στην συνέχεια):

```
# Prepare the data for modeling
X = btc_data[features]  # Features
y = btc_data['Close']  # Target variable (Price)

### DATA SCALING ###
# Standardize the features using StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)  # Scale all data

# Split the data into training and testing sets (80%-20%)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.99, shuffle=False)
```

a. Linear Regression

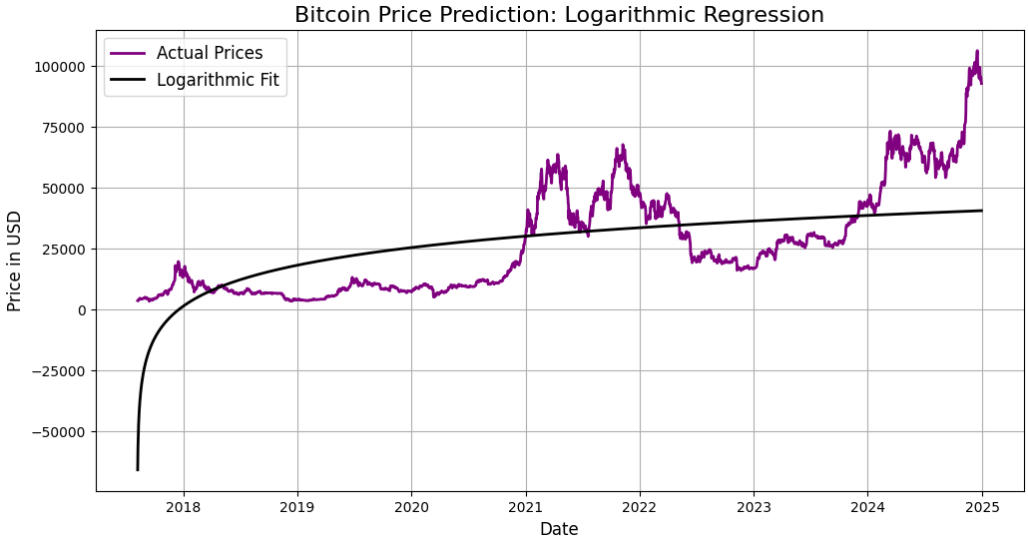
```
# Train a Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)
```




```
Metrics:
Mean Absolute Error (MAE): 1547.2960700997592
Mean Squared Error (MSE): 5211177.781593201
Root Mean Squared Error (RMSE): 2282.800425265687
R-squared (R²): 0.9893016170908435
Mean Absolute Percentage Error (MAPE): 5.39%
```

b. Logarithmic Fit

```
# Define the logarithmic function
def log_function(x, a, b):
    return a + b * np.log(x)
```



```
Metrics:
Mean Absolute Error (MAE): 14678.365063106661
Mean Squared Error (MSE): 305724930.0044564
Root Mean Squared Error (RMSE): 17484.991564323285
R-squared (R²): 0.36932197956097523
Mean Absolute Percentage Error (MAPE): 111.78%
```

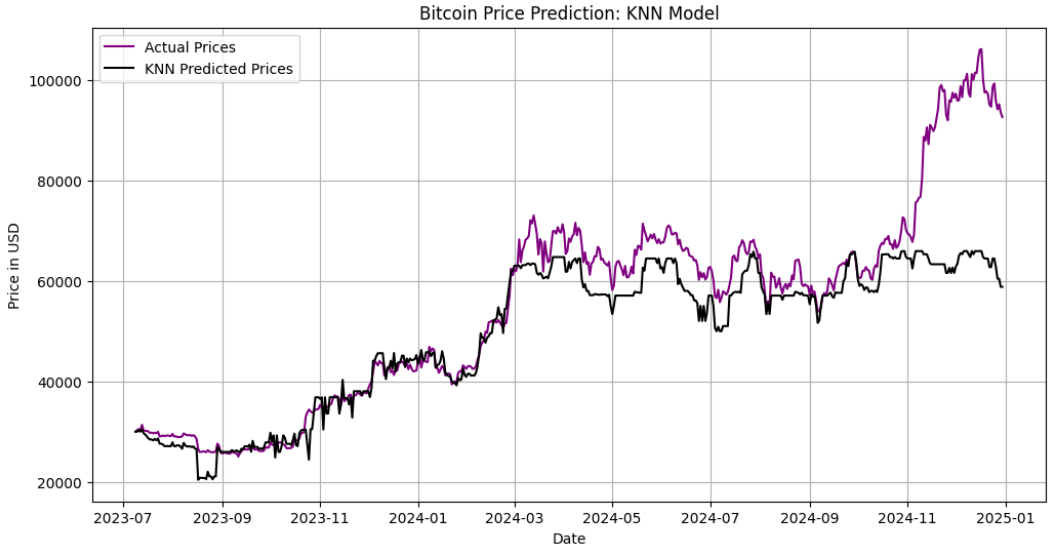
c. *Exponential Fit*

```python
# Define the exponential function
def exponential_function(x, a, b):
    return a * np.exp(b * x)
```


```
Metrics:
Mean Absolute Error (MAE): 10277.124899017663
Mean Squared Error (MSE): 180883289.23219496
Root Mean Squared Error (RMSE): 13449.285826102252
R-squared (R²): 0.6268570090708749
Mean Absolute Percentage Error (MAPE): 52.70%
```

d. *kNN*

```python
# Initialize the KNN Regressor with k=5
knn = KNeighborsRegressor(n_neighbors=5)
```

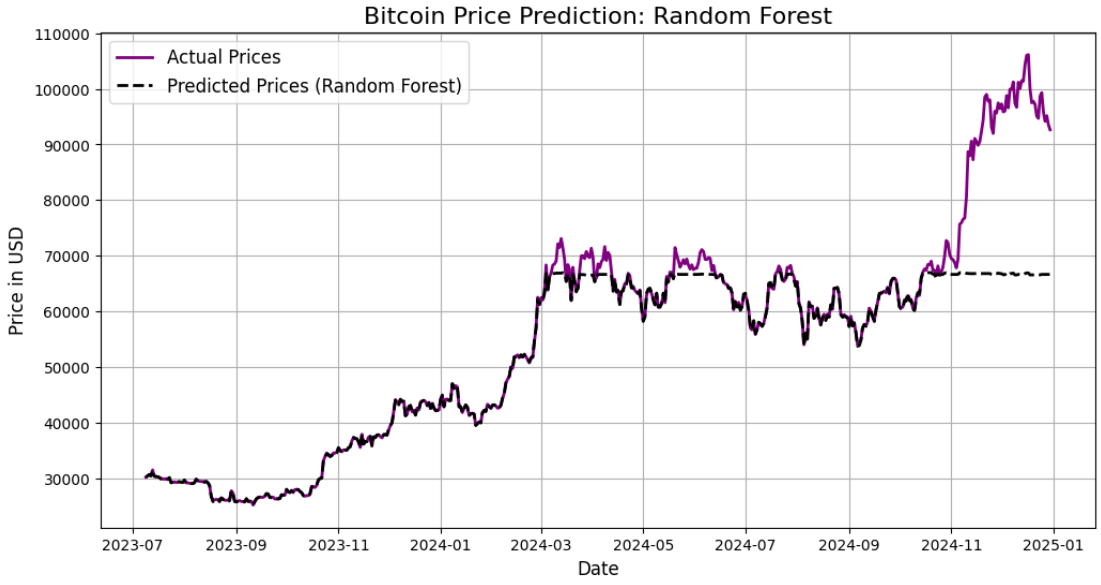



```
Model Evaluation Metrics:
Mean Absolute Error (MAE): 5883.01
Mean Squared Error (MSE): 114373956.93
R-squared (R2): 0.72
```

e. Random Forest

```
# Initialize the Random Forest model
random_forest = RandomForestRegressor(n_estimators=50, random_state=42, max_depth=20)
```

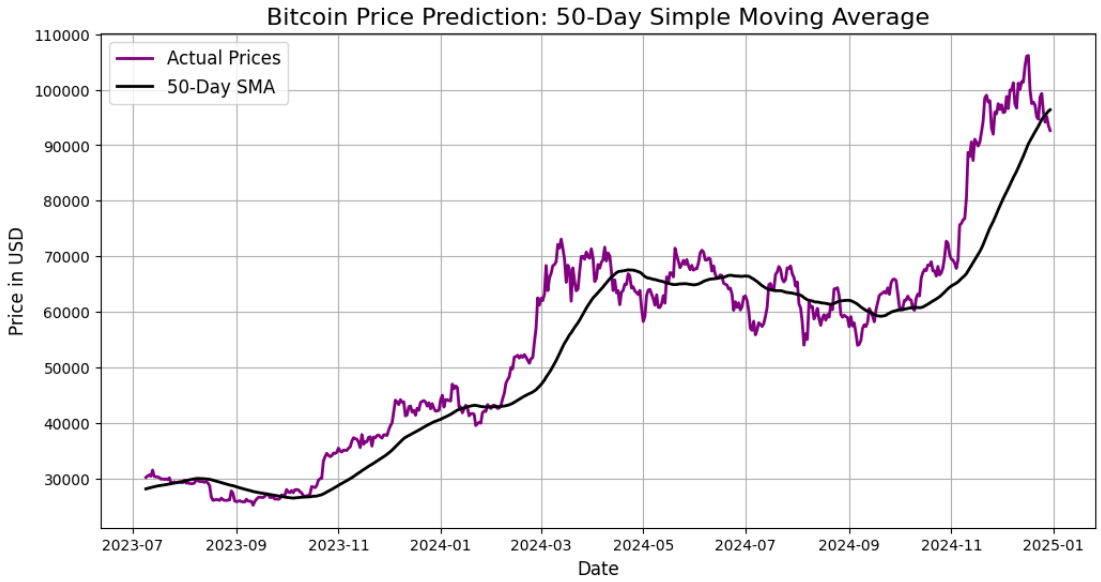
Αρκετά κακό Fit




```
Metrics:
Random Forest - Mean Absolute Error (MAE): 3256.5538138285583
Random Forest - Mean Squared Error (MSE): 85224123.14215964
Random Forest - Root Mean Squared Error (RMSE): 9231.691239537837
```

f. SMA

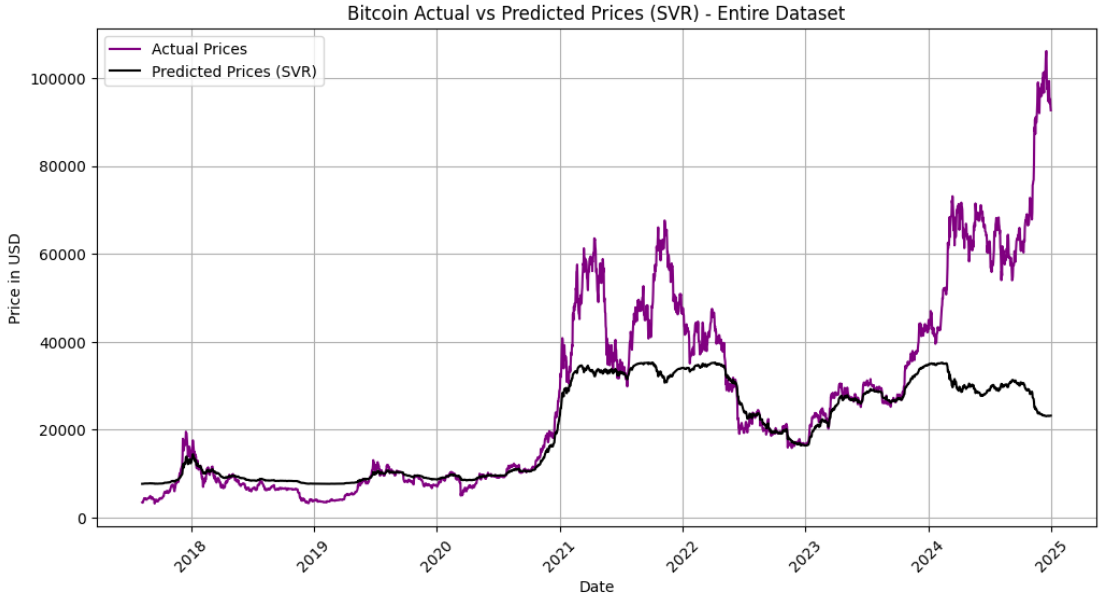
```
# Calculate the Simple Moving Average (SMA)
sma_window = 50
btc_data['SMA'] = btc_data['Close'].rolling(window=sma_window).mean()
```




```
Mean Absolute Error (SMA): 4990.51
Mean Squared Error (SMA): 50186587.65
Root Mean Squared Error (SMA): 7084.25
R-squared (R²): 0.88
Mean Absolute Percentage Error (MAPE): nan%
```

g. SVR

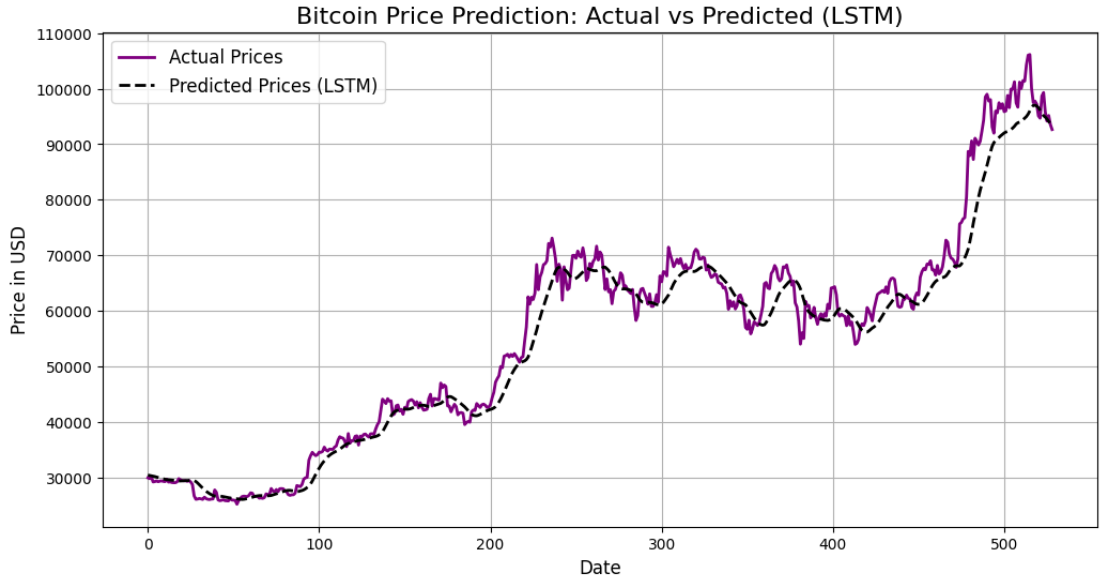
```
# Step 4: Initialize and train the SVR model
svr = SVR(kernel='rbf', C=30, epsilon=0.01)  # Radial Basis Function kernel
```




```
Mean Absolute Error (SVR): 29505.54
Mean Squared Error (SVR): 1323942399.21
Root Mean Squared Error (RMSE) (SVR): 36386.02
R-squared (R²) (SVR): -2.28
Mean Absolute Percentage Error (MAPE) (SVR): 46.63%
```

h. LSTM

```python
# Initialize the LSTM model
lstm_model = Sequential()

# Add LSTM layers
lstm_model.add(LSTM(units=50, return_sequences=True, input_shape=(X_tra
lstm_model.add(Dropout(0.2))  # Dropout for regularization
lstm_model.add(LSTM(units=50, return_sequences=False))
lstm_model.add(Dropout(0.2))

# Add output layer
lstm_model.add(Dense(units=1))

# Compile the model
lstm_model.compile(optimizer='adam', loss='mean_absolute_error')

# Train the model
lstm_model.fit(X_train, y_train, epochs=1, batch_size=32, verbose=1)
```




```
Metrics:
LSTM - Mean Absolute Error (MAE): 2644.44
LSTM - Mean Squared Error (MSE): 14633894.42
LSTM - Root Mean Squared Error (RMSE): 3825.43
LSTM - R-squared (R²): 0.96
LSTM - Mean Absolute Percentage Error (MAPE): 4.41%
```

6.Βελτιστοποίηση Μοντέλων

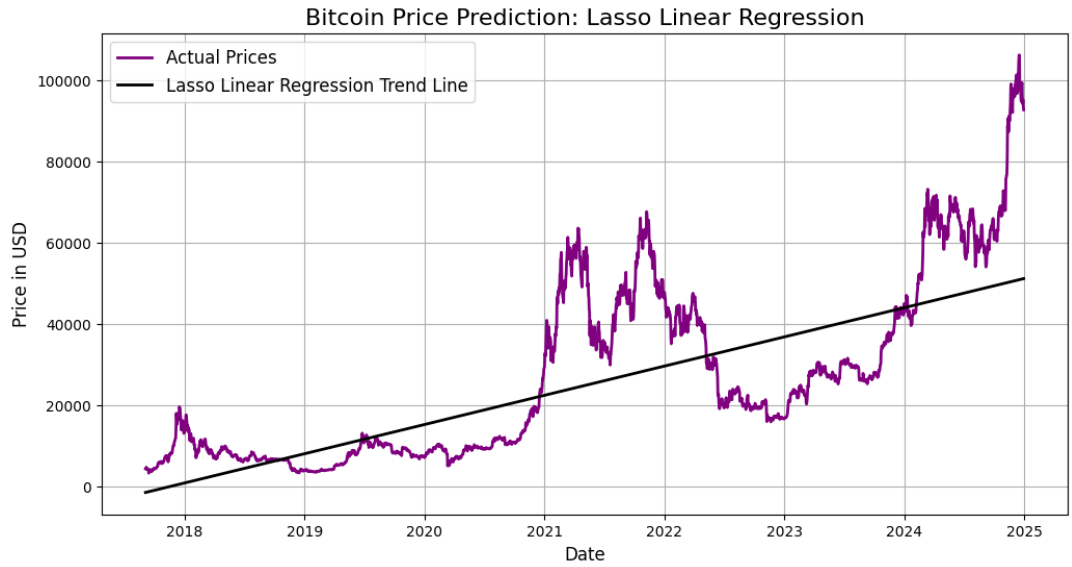
Στο κομμάτι αυτό θα γίνει κυρίως Hyperparameter Tuning(Grid Search, Random Search, Hybrid Search) και Cross Validation(K-Fold, Time Series Split, Leave one out)

Για τα Μοντέλα Linear Regression, Logarithmic Fit και Exponential Fit θα εφαρμοστεί L1(Lasso Regularization) και L2(Ridge Regularization)

Kfold θα γίνει κυρίως 2-5 λόγω των μεγάλων χρονικών διαστημάτων που απαιτούνται για την εκτέλεση για τους περισσότερους κώδικες.

1. ***Linear Regression***

- ***L1***



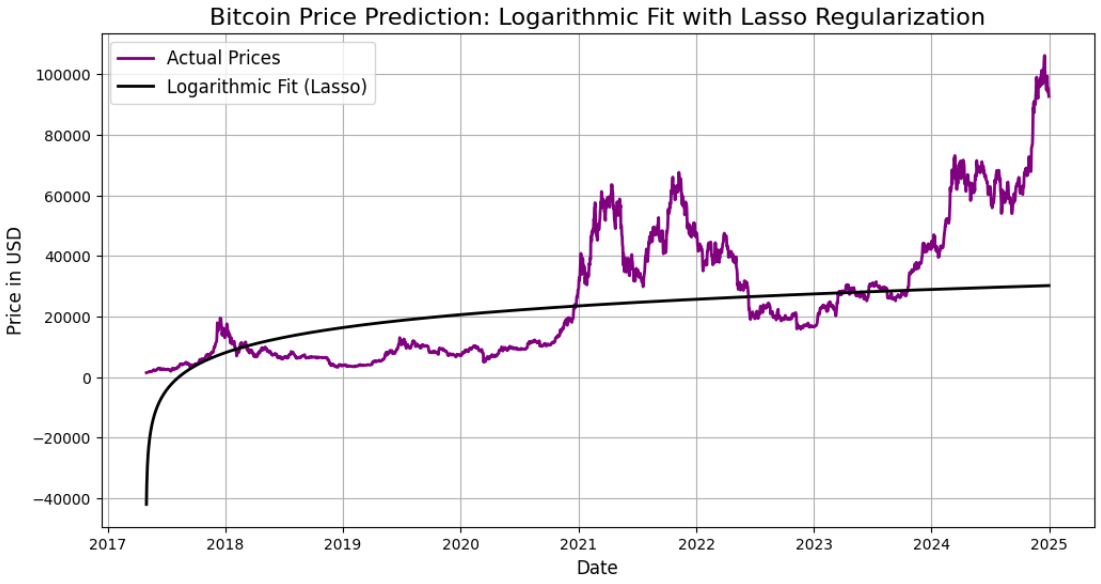

```
Metrics:
Mean Absolute Error: 2439.59
Mean Squared Error: 10483645.77
Root Mean Squared Error: 3237.85
R-squared (R²): 0.98
Mean Absolute Percentage Error (MAPE): 8.18%
```

- ***L2***


```
Unstandardized Metrics:
Mean Absolute Error: 38841.46610403743
Mean Squared Error: 2009893159650.6487
Root Mean Squared Error: 48831.83194618137
```

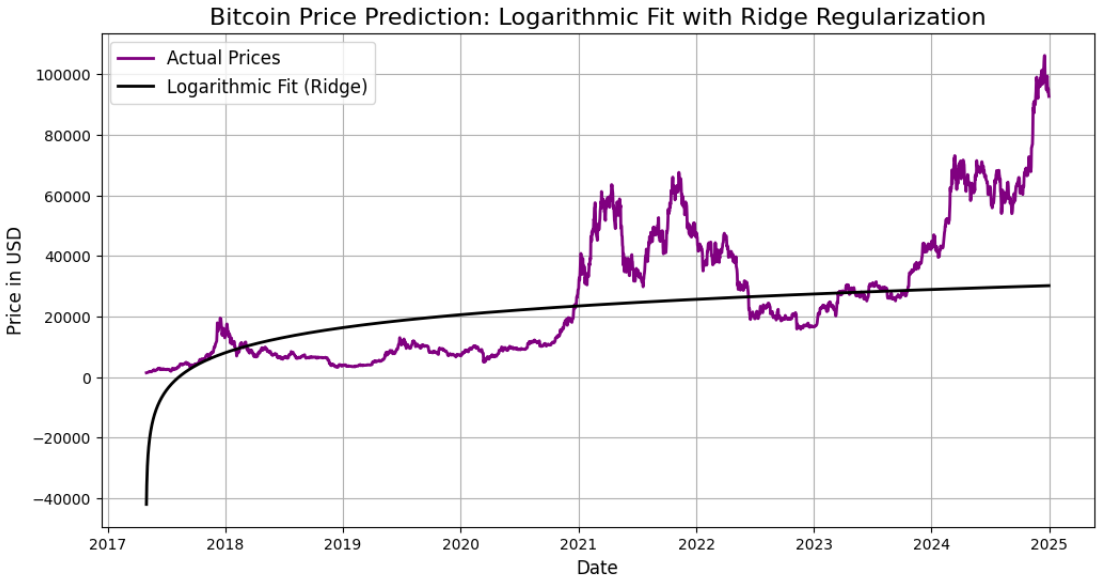
2. Logarithmic Fit

- ***L1***




```
Metrics:
Mean Absolute Error: 25277.919484339884
Mean Squared Error: 1007530688.7673882
Root Mean Squared Error: 31741.623915095905
```

- ***L2***



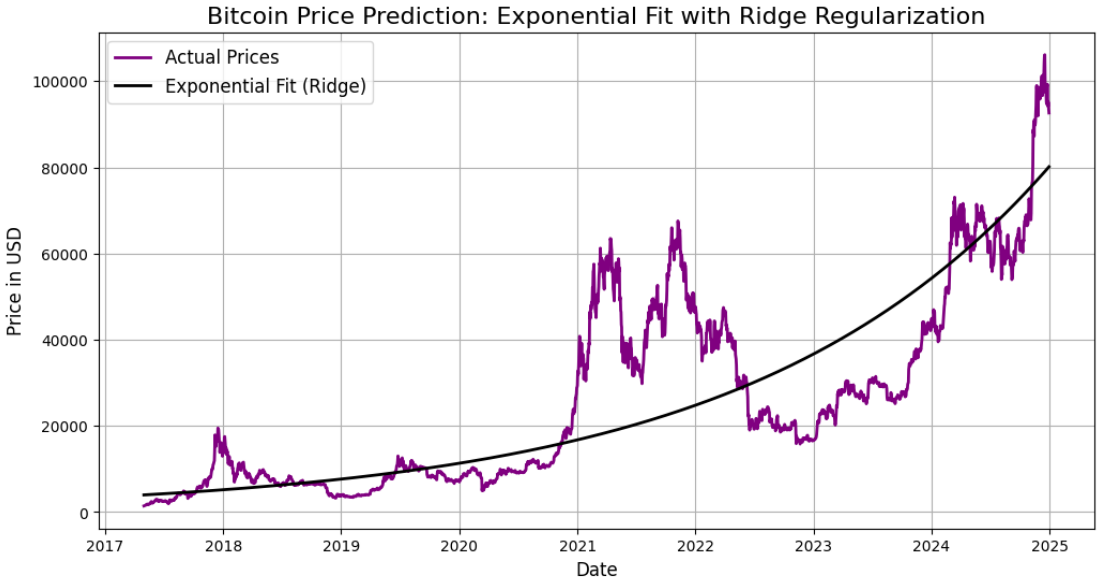

```
Metrics:
Mean Absolute Error: 25281.5305157956
Mean Squared Error: 1007768086.8397433
Root Mean Squared Error: 31745.363233703018
```

3. Exponential Fit

- ***L1***

```
Metrics:
Mean Absolute Error: 10555.481092868991
Mean Squared Error: 164137739.50092798
Root Mean Squared Error: 12811.625170169786
```

- ***L2***



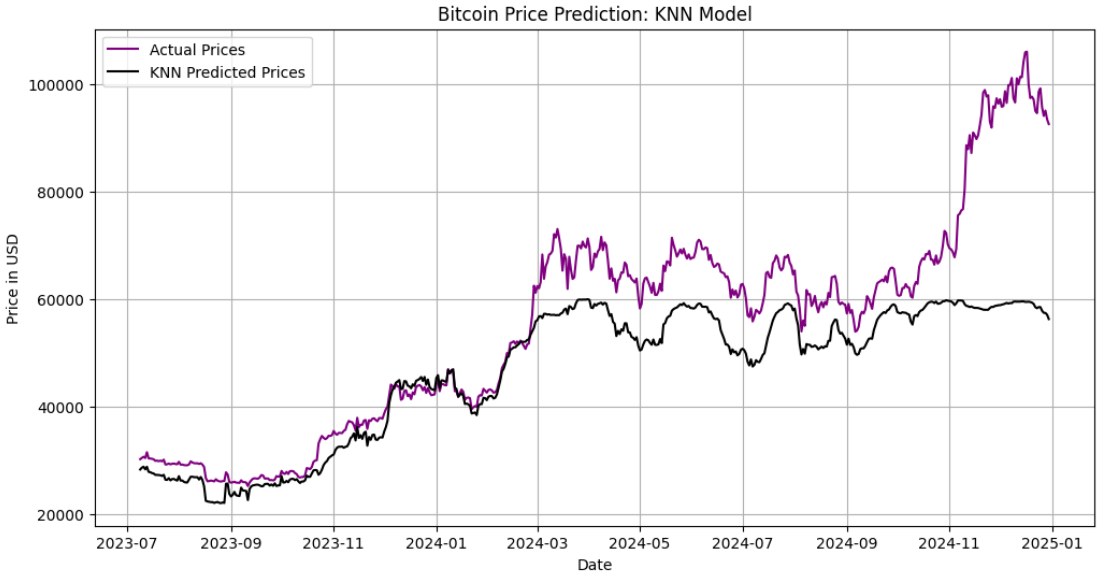

```
Metrics:
Mean Absolute Error: 11603.366596281001
Mean Squared Error: 175264616.06989014
Root Mean Squared Error: 13238.754324704803
```

4. kNN

Hyperparameters:

```python
param_grid = {
    'n_neighbors': list(range(100, 200)),  # Number of neighbors
    'weights': ['uniform', 'distance'],  # Weight function used in prediction
    'p': [1, 2],  # Power parameter for Minkowski metric (1=Manhattan, 2=Euclidean)
    'metric': ['minkowski', 'euclidean', 'manhattan']  # Distance metric
}
```

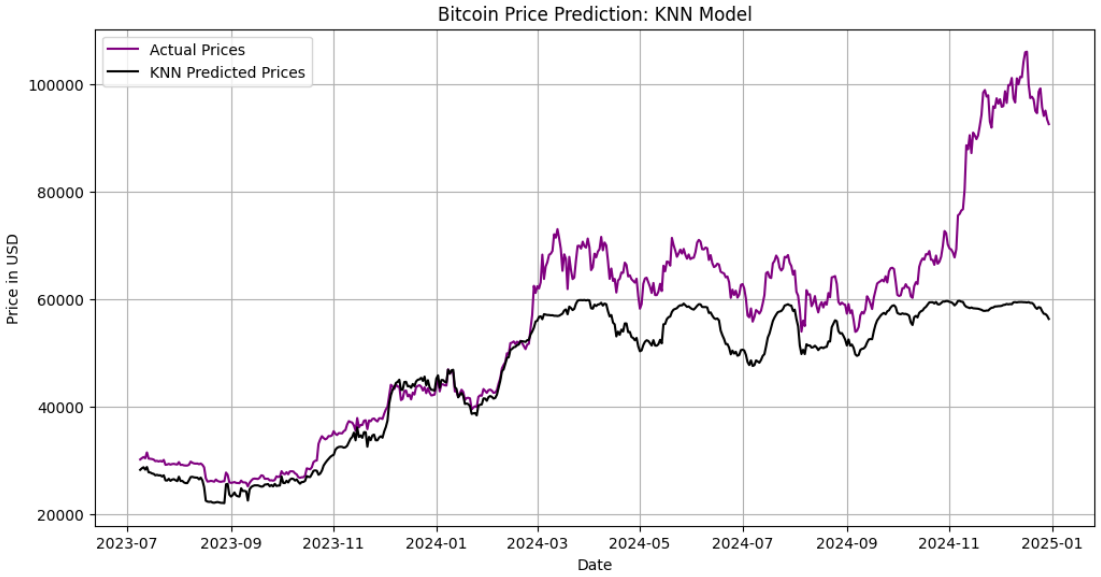
- ***Grid Search***




```
Model Evaluation Metrics:
Mean Absolute Error (MAE): 8636.02
Mean Squared Error (MSE): 176916185.88
R-squared (R2): 0.56
```

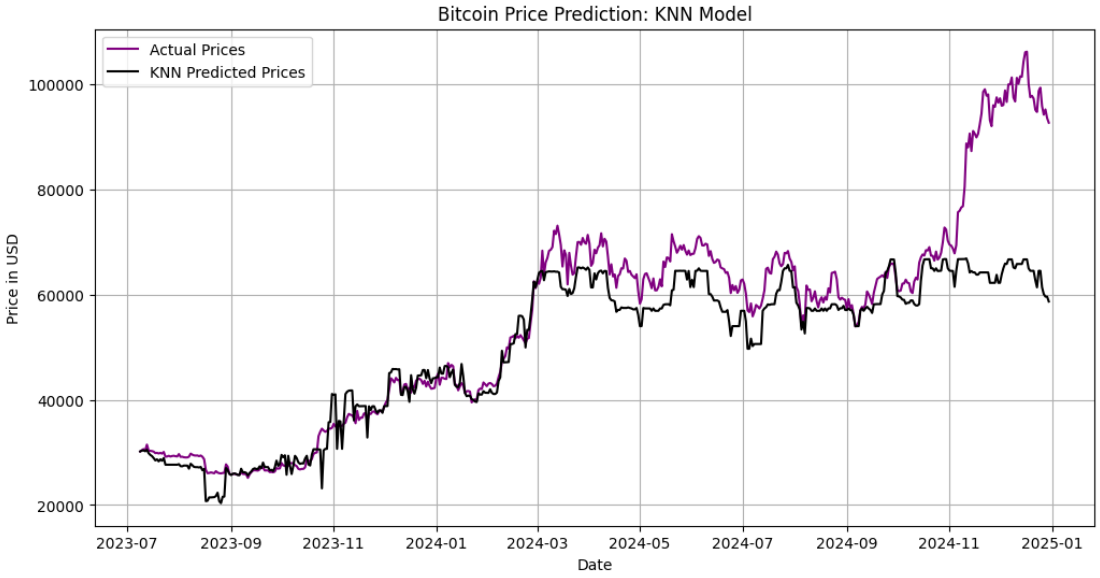
- ***Random Search***

  (Basically the same)




```
Model Evaluation Metrics:
Mean Absolute Error (MAE): 8683.21
Mean Squared Error (MSE): 178309329.40
R-squared (R2): 0.56
```

- ***Hybrid Search***




```
Model Evaluation Metrics:
Mean Absolute Error (MAE): 5878.29
Mean Squared Error (MSE): 112949253.89
R-squared (R2): 0.72
```

K-Fold,TSS,LOOM:

KFold Evaluation

- Mean RMSE: 516.91
- Mean MAE: 295.57

TSS Evaluation

- Mean RMSE: 5727.63
- Mean MAE: 4779.73

LOOCV Evaluation

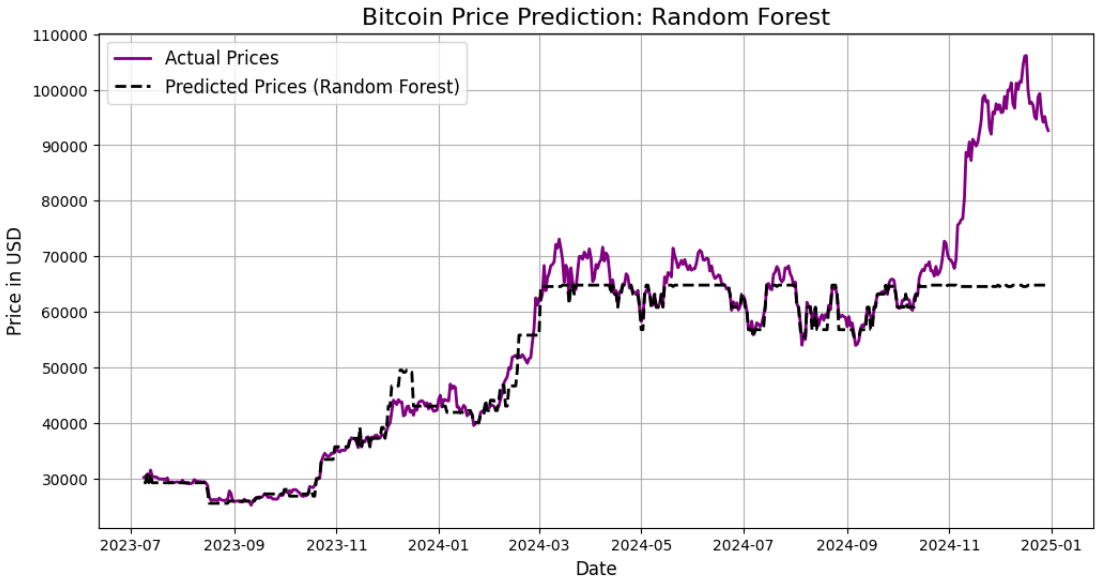
- Mean RMSE: 549.47
- Mean MAE: 549.47

5. *Random Forest*

Hyperparameters:

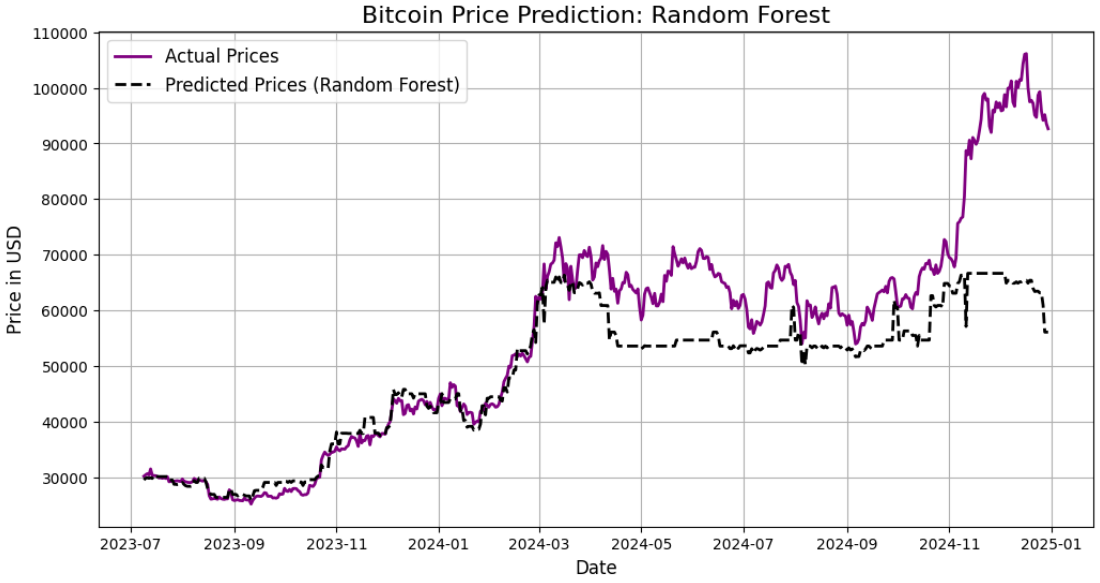
```
param_grid = {
    'n_estimators': [1, 2, 3, 4 ],            # Number of trees in the forest
    'max_depth': [None, 1, 2, 3, 4],            # Maximum depth of the tree
    'min_samples_split': [1, 2, 3, 4],            # Minimum number of samples required to split an internal node
    'min_samples_leaf': [1, 2, 4, 8],            # Minimum number of samples required to be at a leaf node
    'max_features': ['auto', 'sqrt', 'log2'],            # The number of features to consider for splitting at each node
    'bootstrap': [True, False]                        # Whether to use bootstrap samples for building trees
}
```

- ***Grid Search***



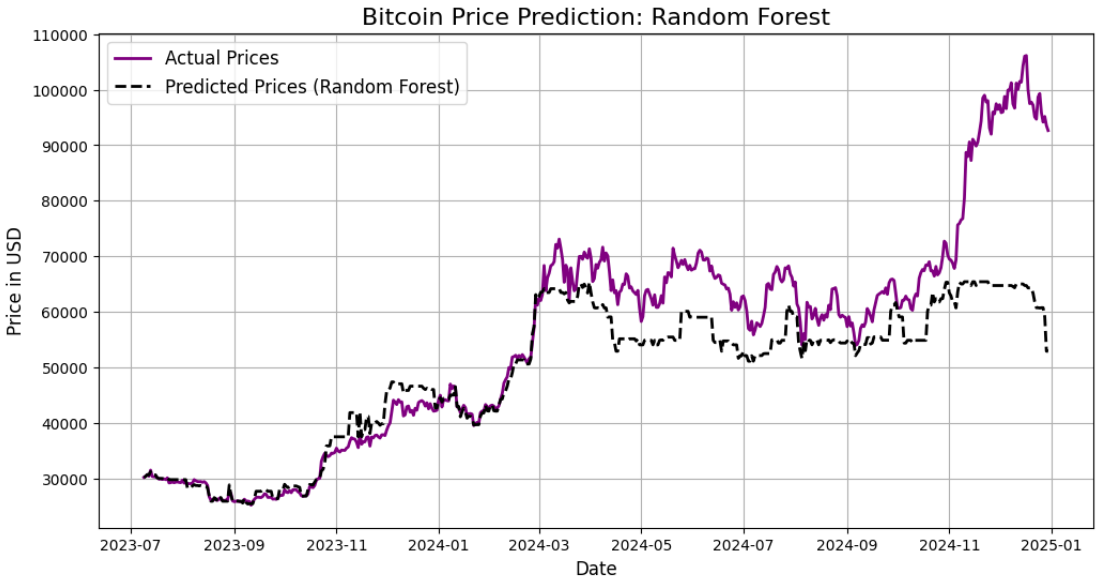

```
Metrics:
Random Forest - Mean Absolute Error (MAE): 4517.762611022634
Random Forest - Mean Squared Error (MSE): 101535256.57632314
Random Forest - Root Mean Squared Error (RMSE): 10076.470442388205
```

- ***Random Search***




```
Metrics:
Random Forest - Mean Absolute Error (MAE): 7317.500992808456
Random Forest - Mean Squared Error (MSE): 133131989.89827831
Random Forest - Root Mean Squared Error (RMSE): 11538.283663451784
```

- ***Hybrid Search(with last hyperparameter n_estimators=50)***




```
Metrics:
Random Forest - Mean Absolute Error (MAE): 7035.696361868454
Random Forest - Mean Squared Error (MSE): 131952922.32735145
Random Forest - Root Mean Squared Error (RMSE): 11487.07631764286
```

K-Fold,TSS,LOOM:

KFold Evaluation

- Mean RMSE: 1362.77
- Mean MAE: 726.76

TSS Evaluation

- Mean RMSE: 5674.78
- Mean MAE: 4640.23

LOOCV Evaluation

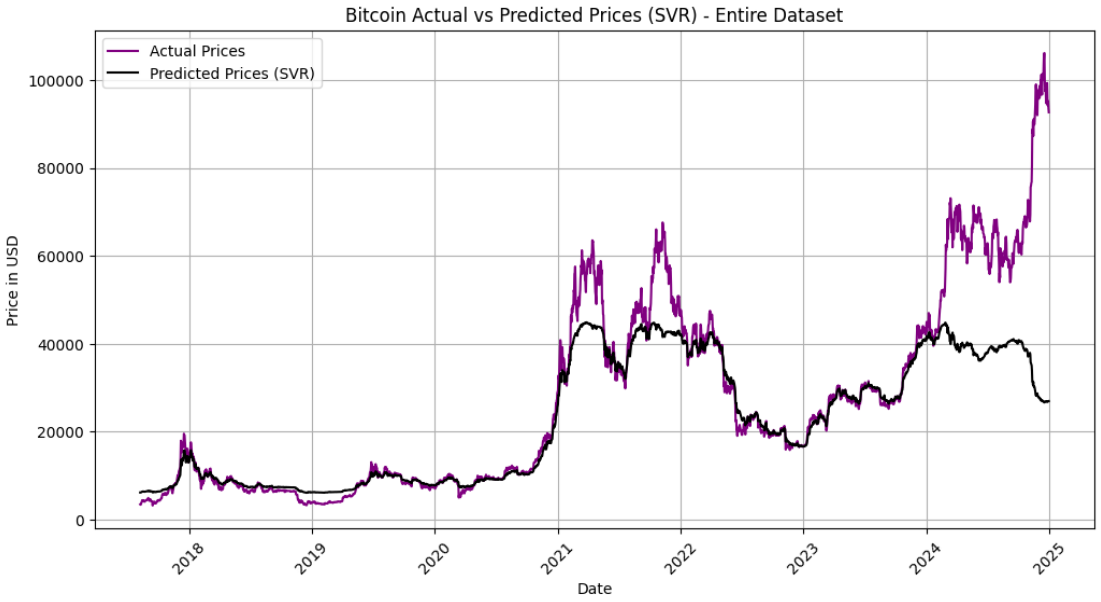
- Mean RMSE: 278.99
- Mean MAE: 271.67

6. SVR

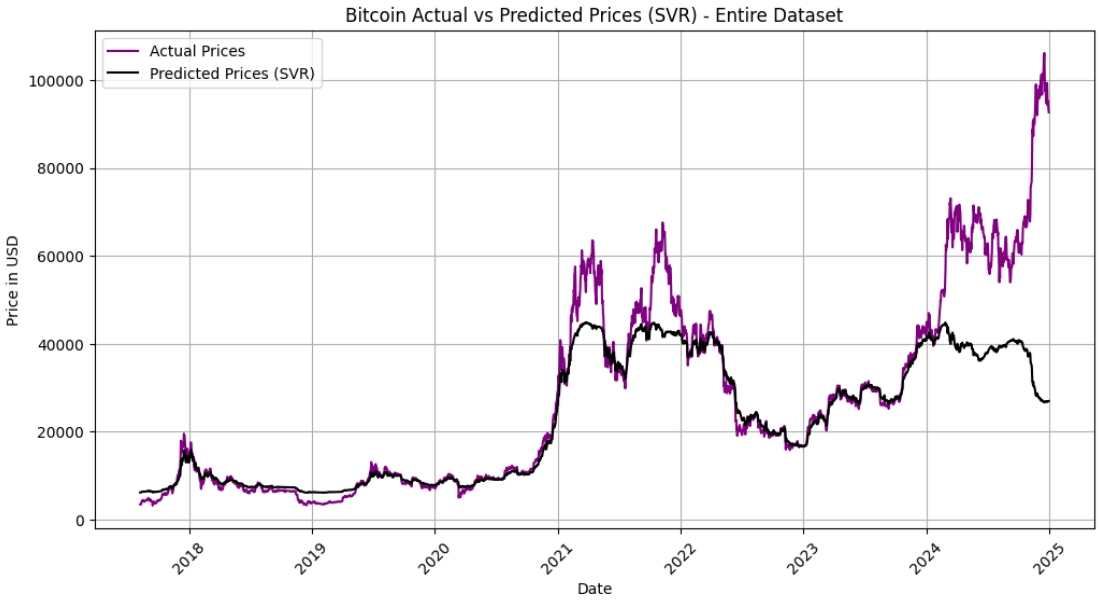
Hyperparameters:

```python
param_grid = {
    'C': [1, 20, 40, 80],                # Regularization parameter
    'epsilon': [0.01, 0.2, 0.4, 2],          # Epsilon in SVR
}
```

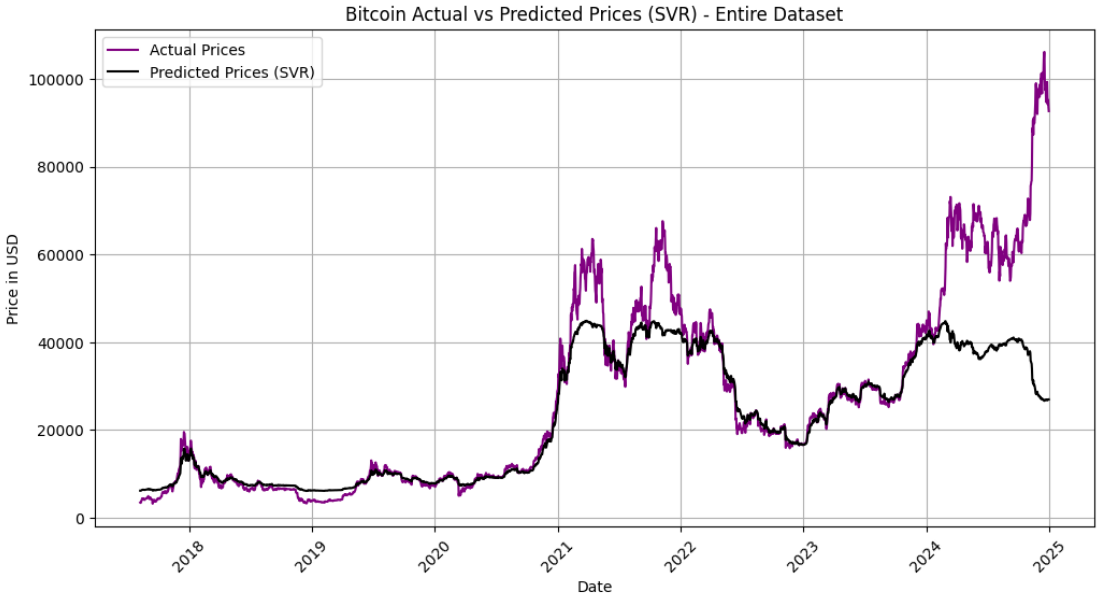
- ***Grid Search***



- ***Random Search***



- ***Hybrid Search***



K-Fold,TSS,LOOM:

Grid

- MAE: 5497.15
- MSE: 165242546.02
- RMSE: 12854.67
- R^2: 0.6591
- MAPE: 132.78%

Random

- MAE: 5497.15
- MSE: 165242546.02
- RMSE: 12854.67
- R^2: 0.6591
- MAPE: 132.78%

Hybrid

- MAE: 5497.15
- MSE: 165242546.02
- RMSE: 12854.67
- R^2: 0.6591
- MAPE: 132.78%

KFold Evaluation

- Mean RMSE: 1905.30
- Mean MAE: 901.23

TSS Evaluation

- Mean RMSE: 5248.95
- Mean MAE: 4348.00

7. LSTM

K-Fold Cross Validation:

- **Average RMSE**: 2100.46
- **Average MAE**: 1600.79

Time Series Split (TSS):

- **Average RMSE**: 2577.24
- **Average MAE**: 1624.19

7.Γενική Ανάλυση – Επιπλέον Βελτιστοποιήσεις

Το κομμάτι αυτό θα χωριστεί σε 4 υπό-ενότητες: Τεχνική Ανάλυση, Θεμελιώδη Ανάλυση, Ψυχολογική Ανάλυση, Τελικά Αποτελέσματα

Θα χρησιμοποιηθούν σταθμισμένοι τύποι και σκορ ώστε να μετρήσουν την επιπλέον πιθανή μελλοντική αύξηση της τιμής για κάθε μοντέλο μας(θα προστεθεί προς το τέλος)

Τα σκορ και οι τύποι βγαίνουν με βάση αυτά που πιστεύουμε πιο σημαντικά όπου και θα δοθεί μεγαλύτερη βάση

I. Τεχνική Ανάλυση

Χρησιμοποιώντας το W διάγραμμα από το yahoo finance πάλι, αξιοποιώντας τους τεχνικούς δείκτες RSI,ROC,Williams%,EMA100 μπορούμε να κάνουμε μία μικρή ανάλυση.(προφανώς δείγμα ανάλυσης όπως και στα επόμενα)
Θα υπολογιστούν συνολικά buy και sell signals.

- ***RSI***

```python
def calculate_rsi(data, window=14):
    delta = data['Close'].diff()
    gain = delta.where(delta > 0, 0)
    loss = -delta.where(delta < 0, 0)
    avg_gain = gain.rolling(window=window, min_periods=1).mean()
    avg_loss = loss.rolling(window=window, min_periods=1).mean()
    rs = avg_gain / (avg_loss + 1e-10)
    rsi = 100 - (100 / (1 + rs))
    return rsi
```

- ***ROC***

```python
def calculate_roc(data, column, period=14):
    # ROC = (Current Price - Price N periods ago) / Price N periods ago * 100
    roc = (data[column] - data[column].shift(period)) / data[column].shift(period) * 100
    return roc
```

- ***Williams%***

```
def calculate_williams(data, period=14):
    # Highest high and lowest low over the period
    highest_high = data['High'].rolling(window=period).max()
    lowest_low = data['Low'].rolling(window=period).min()

    # Williams %R = (Highest High - Close) / (Highest High - Lowest Low) * -100
    williams_r = ((highest_high - data['Close']) / (highest_high - lowest_low)) * -100
    return williams_r
```

- ***EMA100***

```
def calculate_ema(data, period=100):
    # Calculate the Exponential Moving Average
    ema = data['Close'].ewm(span=period, adjust=False).mean()
    return ema
```

```
# Weights for each indicator
w_rsi = 0.1
w_roc = 0.25
w_williams = 0.15
w_ema100 = 0.5
```

```
Total Buy signals: 3975
Total Sell signals: 3767

Weighted Buy signals: 1306.65
Weighted Sell signals: 1070.3

Weighted Percentage Buy signals: +54.971707440207%
Weighted Percentage Sell signals: -45.028292559793016%

Meaning +9.943414880413997% increase in our regression Model's Price Prediction
9.943414880413997
```

II. Θεμελιώδης Ανάλυση

- ***Halvings Score***

```
def halving_score(halving_date, current_date):
    time_difference = (current_date - halving_date).days

    if 0 <= time_difference <= 365:  # 1 year
        return 1
    elif 365 < time_difference < 730:  # Between 1 and 2 years

        return -1
    else:  # Negative time difference (halving in the future)
        return 0
```

- ***Interest Rates***

> - **September 2024**: The Federal Reserve initiated its easing cycle with a 50 basis point (0.50%) reduction in the federal funds rate. WIKIPEDIA
> - **December 18, 2024**: The Federal Open Market Committee (FOMC) lowered the target range for the federal funds rate by 25 basis points (0.25%), setting it to 4.25%–4.50%. HOME

```
ir_value = (0.25 + 0.5)*3  # Big Inflation??
print(f"Value: {ir_value}")
```

- ***Inflation***

December 2024, the United States reported an annual inflation rate of 2.9%,

```
inflation=2.9
i_value = 1 if inflation > 2.5 else 0
print(f"Value: {i_value}")
```

- ***Adoption of BTC***

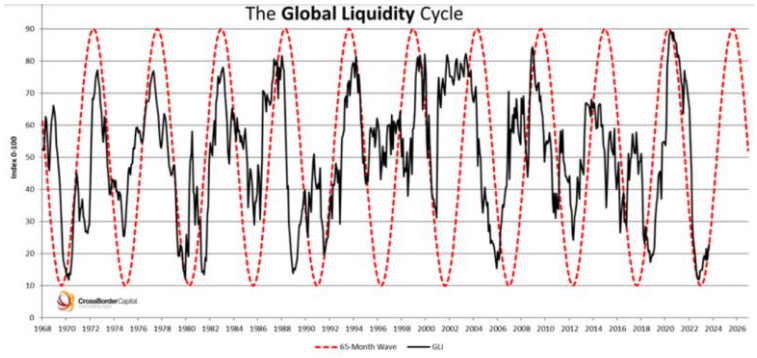
Blackrock, Fidelity, Grayscale, Microstrategy. Pension Funds and Institutional Investments Approval of Bitcoin ETFs Performance of Crypto Hedge Funds

```
[ ] a_value = 2
    print(f"Value: {a_value}")
```

- ***M2SL***


```
]  #Incredible Bounce and increase for M2SL

   m_value = 1
   print(f"Value: {m_value}")
```

- ***Global Liquidity***




```
#Expected increase for Global Liquidity for the next 1-2 years

g_value = 1
print(f"Value: {g_value}")
```

Make a FA Final Assumption

```
fa_value=(h_value+ir_value+i_value+a_value+m_value+g_value)*2
print(f"Fundamental Indication: {fa_value}")

#increase-decrease in model = score*3%
if fa_value>0:
    print(f"Meaning a {fa_value}% Increase in Models's Price Prediction")
else:
    print(f"Meaning a {fa_value}% Decrease in Models's Price Prediction")

print(fa_value)
```

```
Fundamental Indication: 16.5
Meaning a 16.5% Increase in Models's Price Prediction
16.5
```

III. Ψυχολογική Ανάλυση

- ***Fear & Greed Index***

```
Fear and Greed Index: 75
Classification: Greed
Timestamp: 1736985600
Value: 0
```

- ***Geopolitical News***


```
pa_value=fg_value[3]+news_value+ge_value
print(f"Pshycological Indication: {pa_value}/3")

#increase-decrease in model = score*3%
if pa_value>0:
    print(f"Meaning a {pa_value*3}% Increase in Models's Price Prediction")
else:
    print(f"Meaning a {pa_value*3}% Decrease in Models's Price Prediction")

pa_value=pa_value*3
print(pa_value)
```

```
Pshycological Indication: 2/3
Meaning a 6% Increase in Models's Price Prediction
6
```

IV. Τελικά Αποτελέσματα(Συγχώνευση Αποτελεσμάτων σε έναν weighted τύπο)

Τελικά Ορίζουμε Weights:

ta_weight = 0.2
fa_weight = 0.6
pa_weight = 0.2

και υπολογιζουμε :
Total_Value = (ta_value * ta_weight + fa_value * fa_weight + pa_value * pa_weight)*2

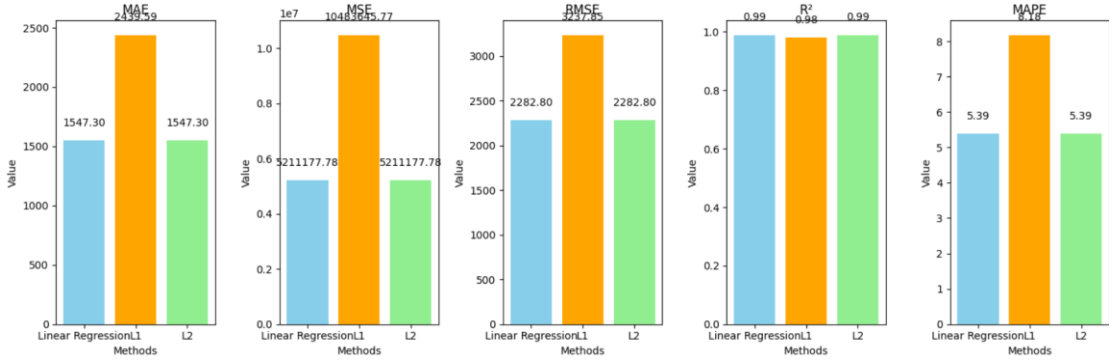
Επομένως θα έχουμε 26% επιπλέον αύξηση στο μοντέλο(αφού όλα δείχνουν θετικά)
Total Value: 26.1773659521656
Therefore 26.1773659521656 increase or decrease in your model.

8.Αξιολογήσεις Μοντέλων

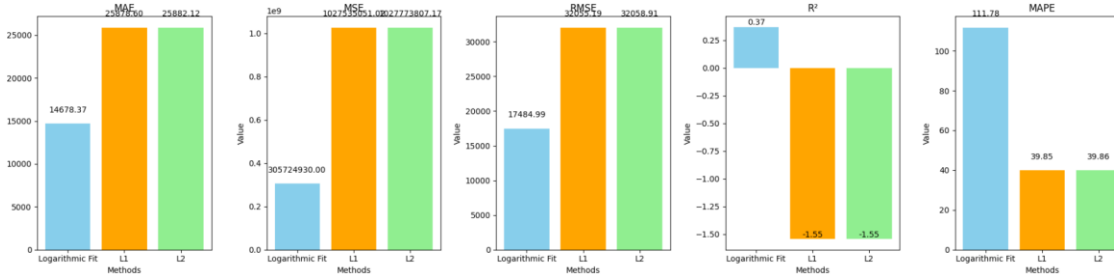
Οι Αξιολογήσεις των Μοντέλων Γίνονται με Βάση τις Μετρικές που προαναφέραμε (rmse,mse,mae,mape,r^2):

Μαζέψαμε για Κάθε Μοντέλο τις Μετρικές και φτιάξαμε Ραβδογράμματα για την καλύτερη απεικόνιση της σχέσης μεταξύ των μετρικών και των μοντέλων ώστε να επιλέξουμε τα καλύτερα:

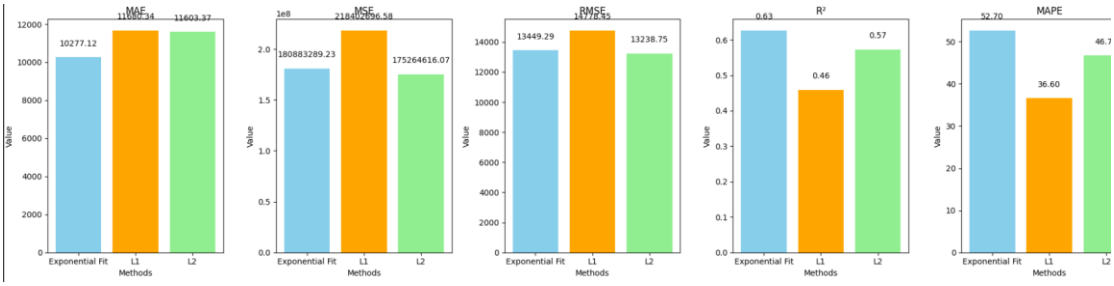
1) Linear Regression



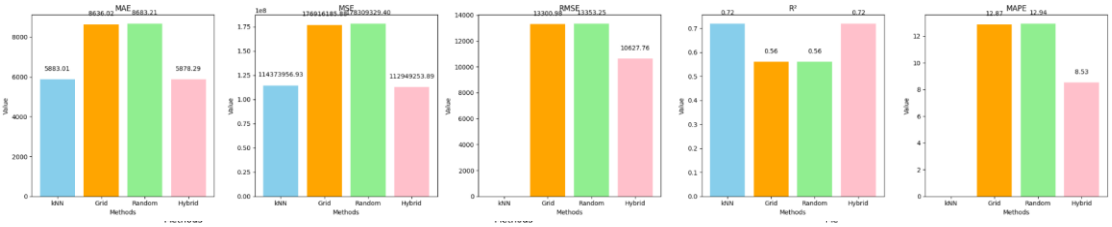
2) Logarithmic Fit

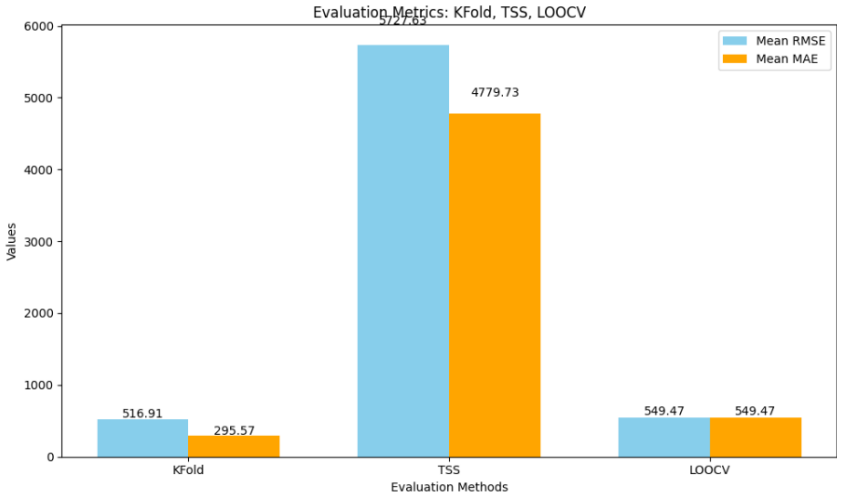


3) Exponential Fit

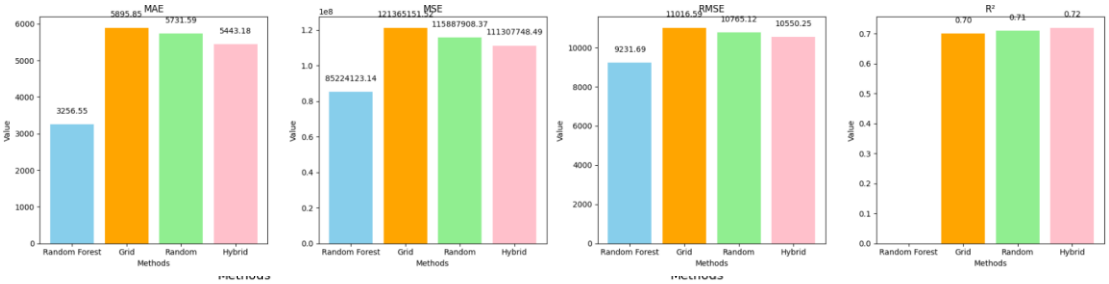


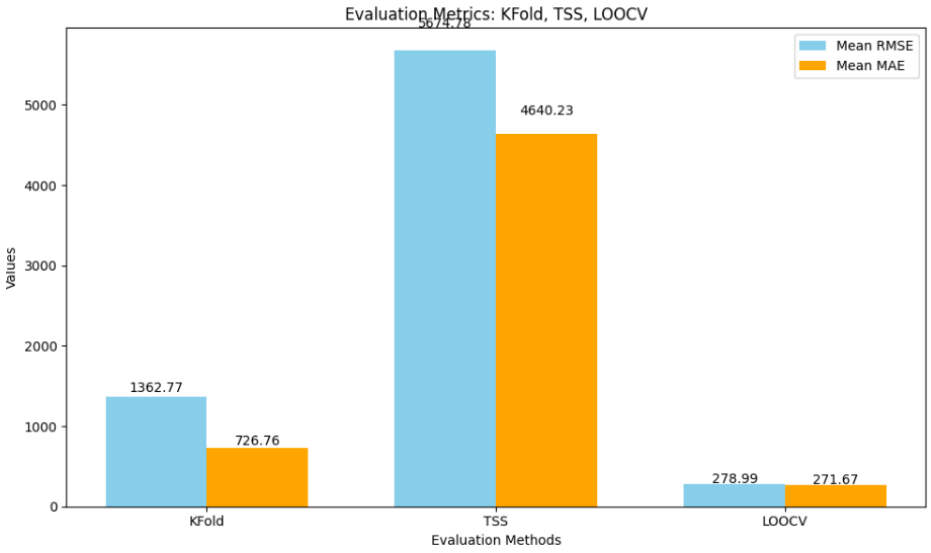
4) kNN



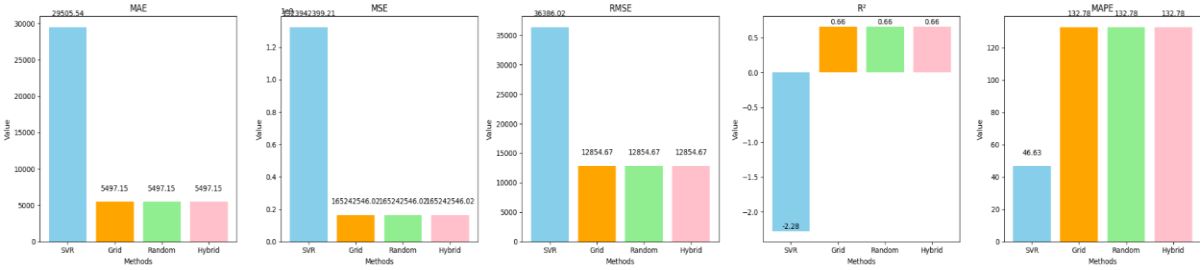


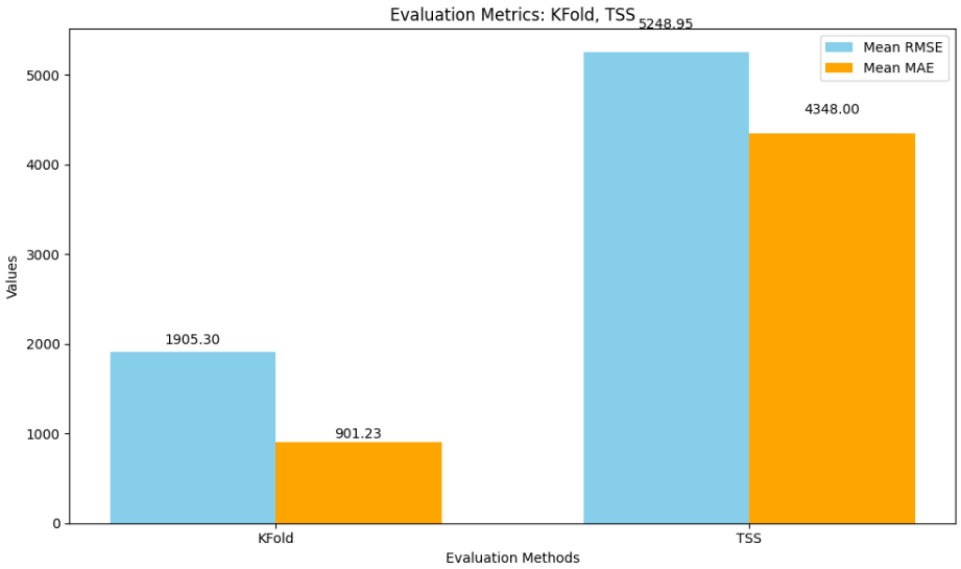
5) Random Forest



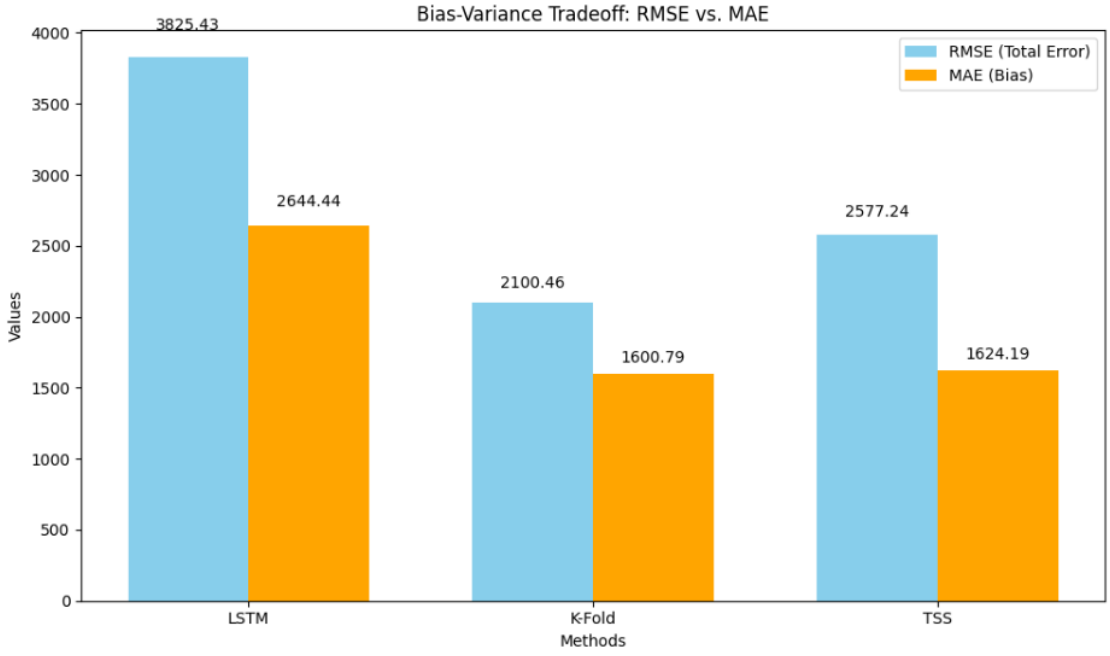


6) SVR


MAE
MSE
RMSE
R²
MAPE
SVR
Grid
Random
Hybrid
Methods
Value

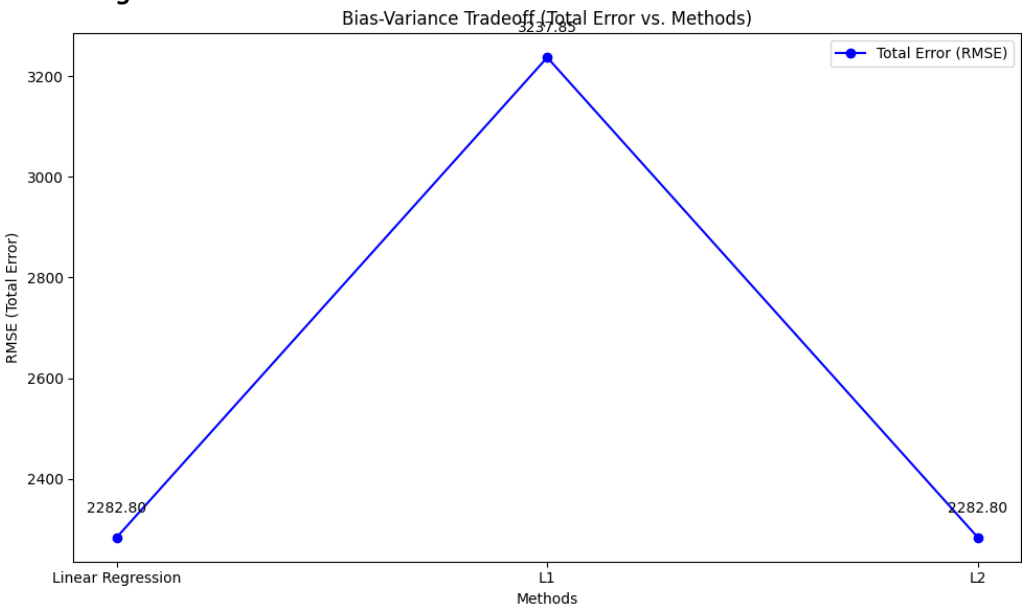

Evaluation Metrics: KFold, TSS
Mean RMSE
Mean MAE
5248.95
4348.00
1905.30
901.23
Values
KFold
TSS
Evaluation Methods

7) LSTM

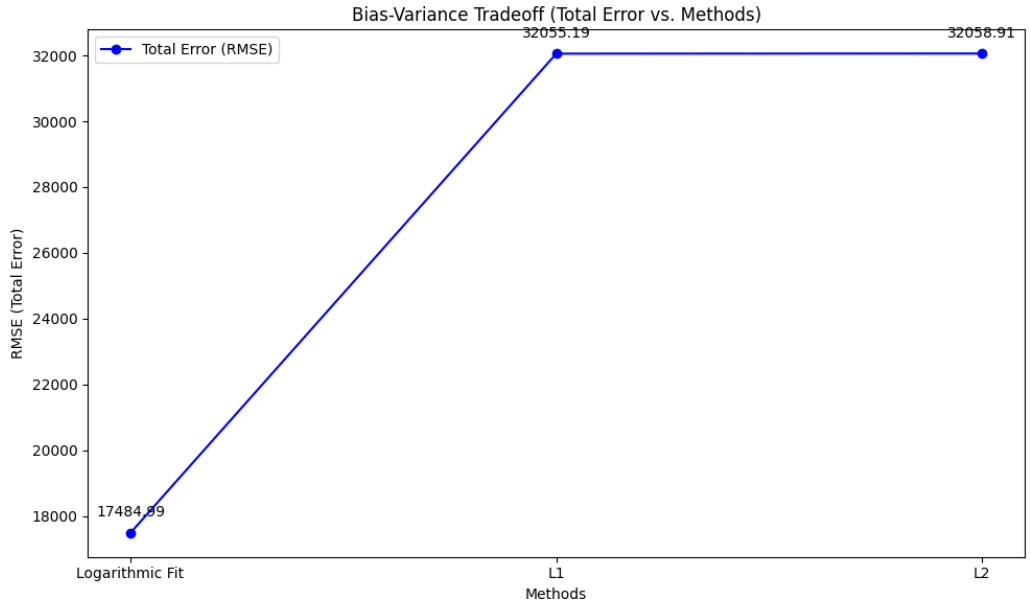

Bias-Variance Tradeoff: RMSE vs. MAE
RMSE (Total Error)
MAE (Bias)
3825.43
2644.44
2100.46
1600.79
2577.24
1624.19
Values
LSTM
K-Fold
TSS
Methods

Επίσης Φτιάξαμε Διαγραμματικές Αποικονίσεις μεταξύ variance και bias tradeoff:

i. Linear Regression

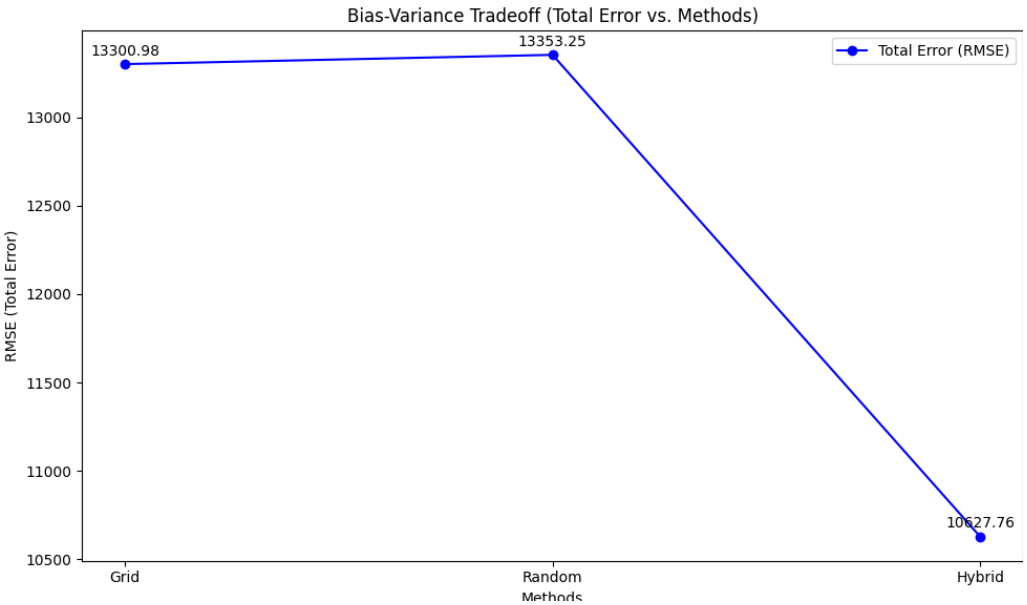


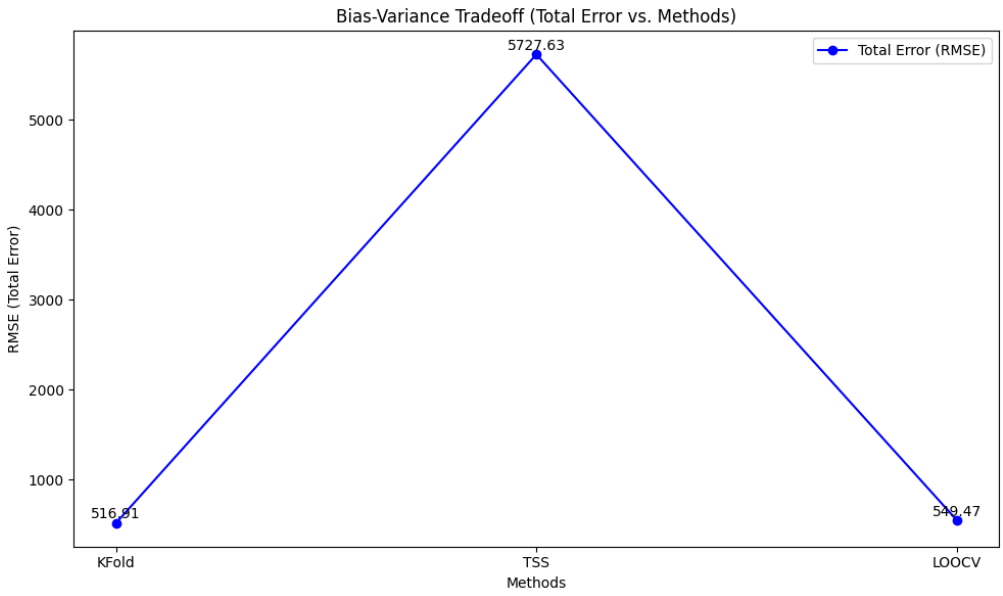
ii. Logarithmic Fit



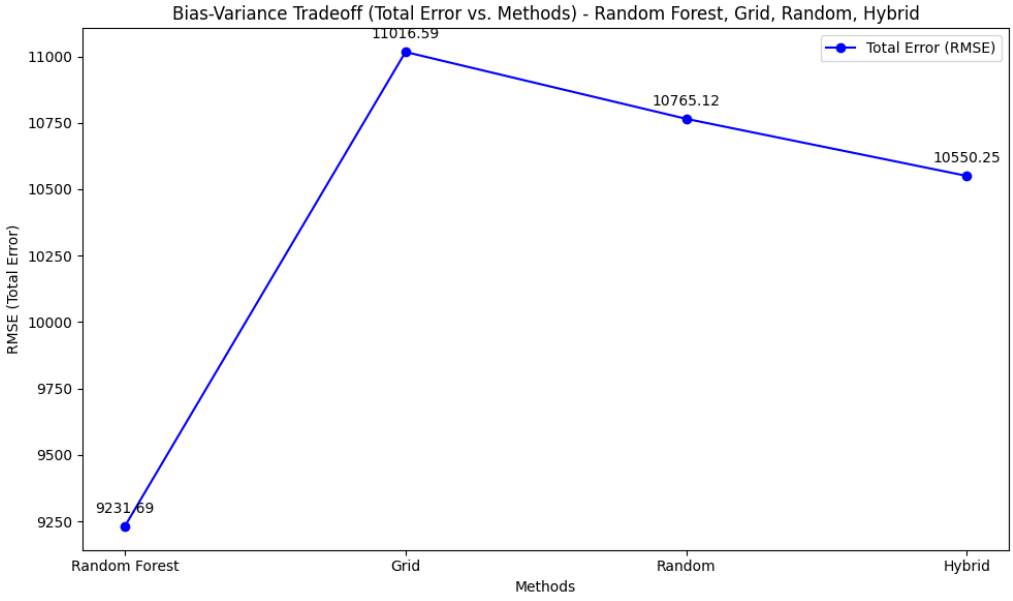
iii. Exponential Fit

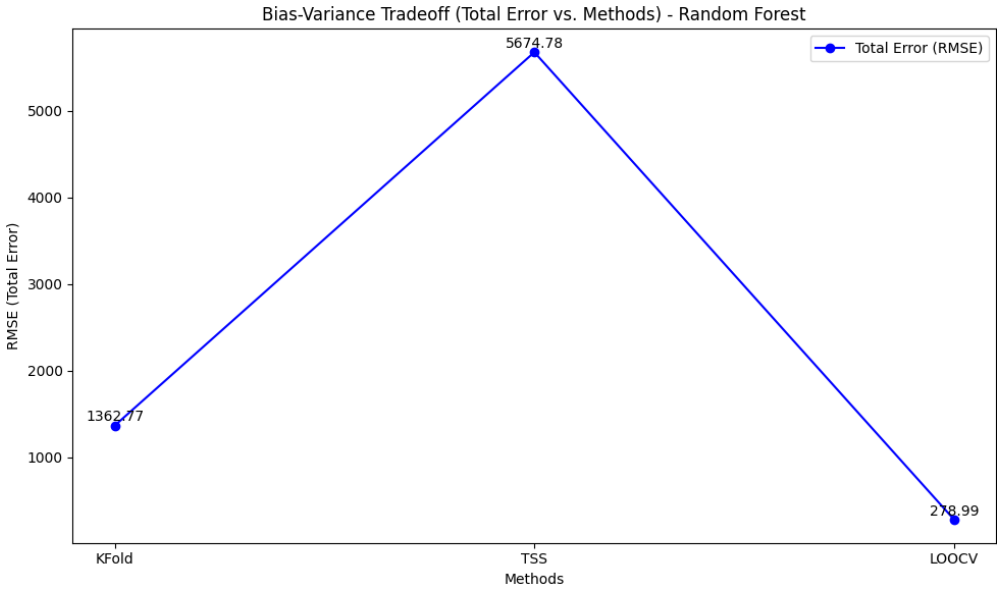
iv. kNN



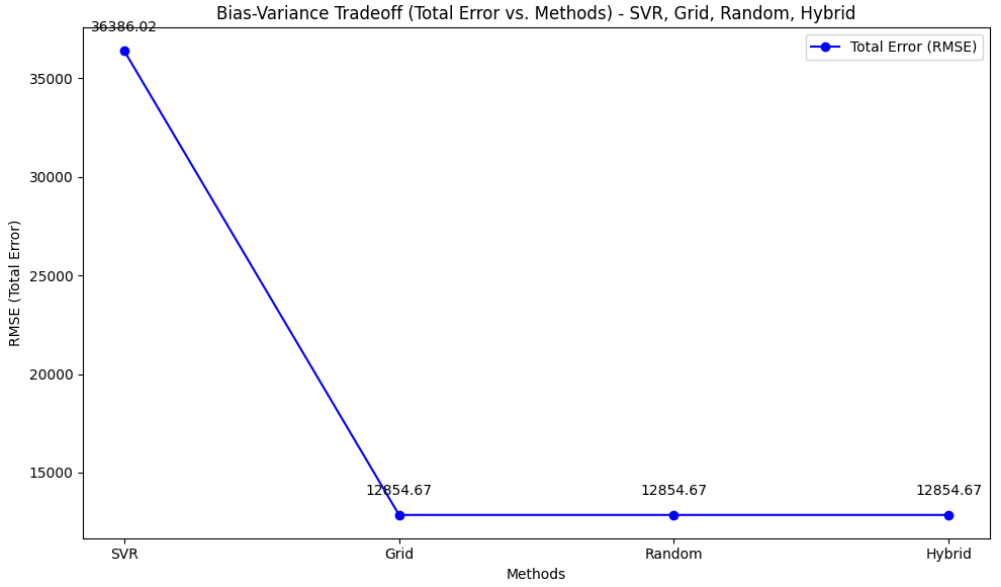


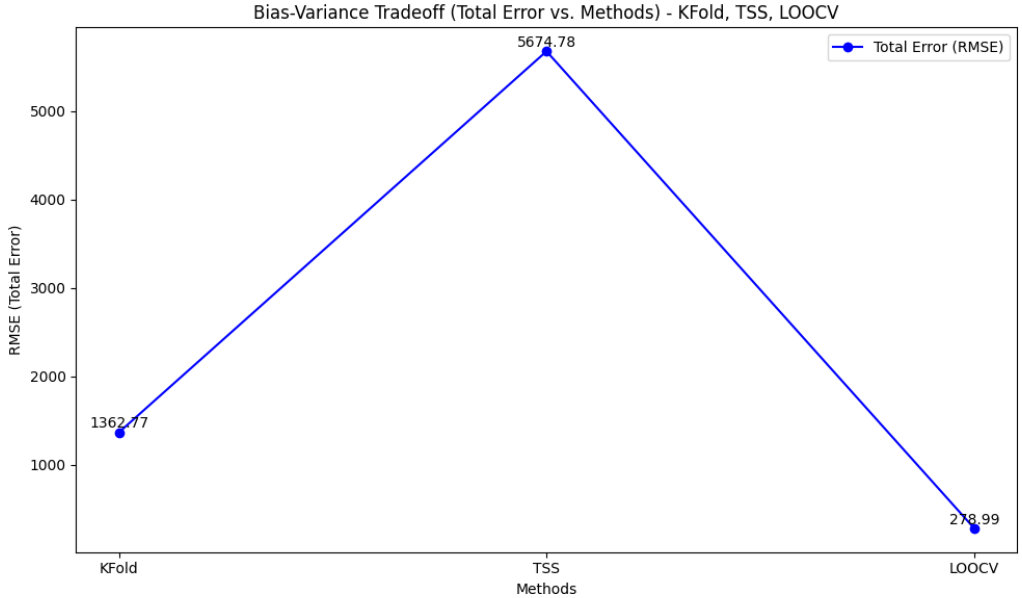
v. Random Forest



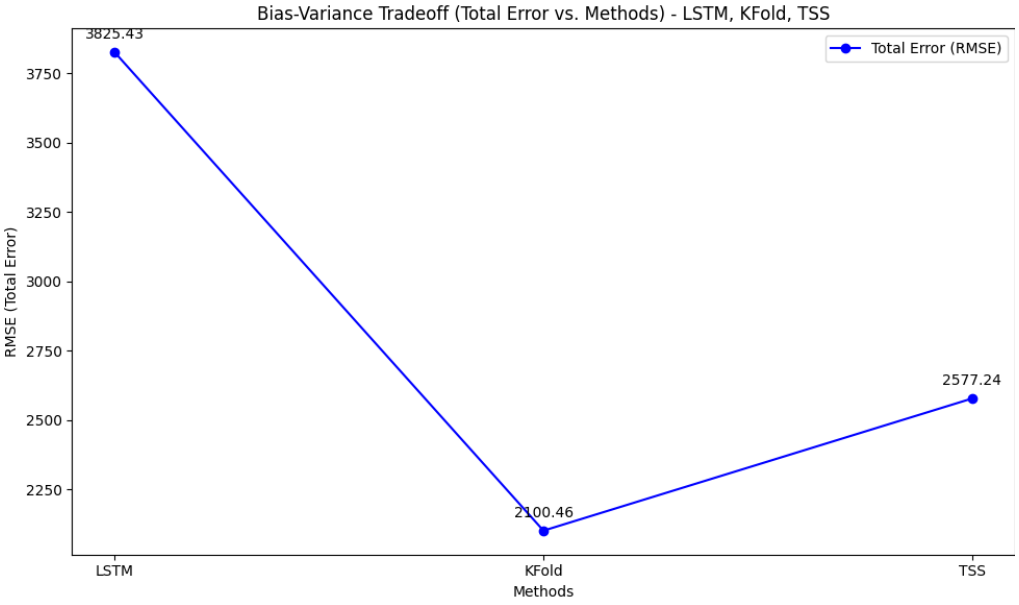

Bias-Variance Tradeoff (Total Error vs. Methods) - Random Forest
Total Error (RMSE)
5674.78
1362.77
278.99
RMSE (Total Error)
5000
4000
3000
2000
1000
KFold
TSS
LOOCV
Methods

vi. SVR


Bias-Variance Tradeoff (Total Error vs. Methods) - SVR, Grid, Random, Hybrid
Total Error (RMSE)
36386.02
12854.67
12854.67
12854.67
RMSE (Total Error)
35000
30000
25000
20000
15000
SVR
Grid
Random
Hybrid
Methods


Bias-Variance Tradeoff (Total Error vs. Methods) - KFold, TSS, LOOCV
Total Error (RMSE)
5674.78
1362.77
278.99
RMSE (Total Error)
5000
4000
3000
2000
1000
KFold
TSS
LOOCV
Methods

vii. LSTM


Bias-Variance Tradeoff (Total Error vs. Methods) - LSTM, KFold, TSS
Total Error (RMSE)
RMSE (Total Error)
3825.43
2100.46
2577.24
LSTM
KFold
TSS
Methods

9. Κάνοντας Προβλέψεις(Τα Καλύτερα Μοντέλα)

Αφού Υπολογίσαμε Όλες τις μετρικές, Δείκτες κτλπ. και βρήκαμε τα Καλύτερα Μοντέλα, τώρα θα δείξουμε τις Προβλέψεις των Μοντέλων μας(Στο κομμάτι αυτό θα προστεθεί η Γενικά Ανάλυση που κάναμε πριν).

Οι προβλέψεις γίνονται στην τελευταία τιμή του BTC του test data(ώστε να δούμε πόσο κοντά πέσανε).

BTC Price Εκεί που Προβλέπουμε: 93500.00

Προβλέψεις(Καλύτερων Μοντέλων):

L2 Linear Regression = 106338.24

Logarithmic Fit = 38029.59

L2 Exponential Fit = 100977.15

kNN(Hybrid Search) = 74214.0

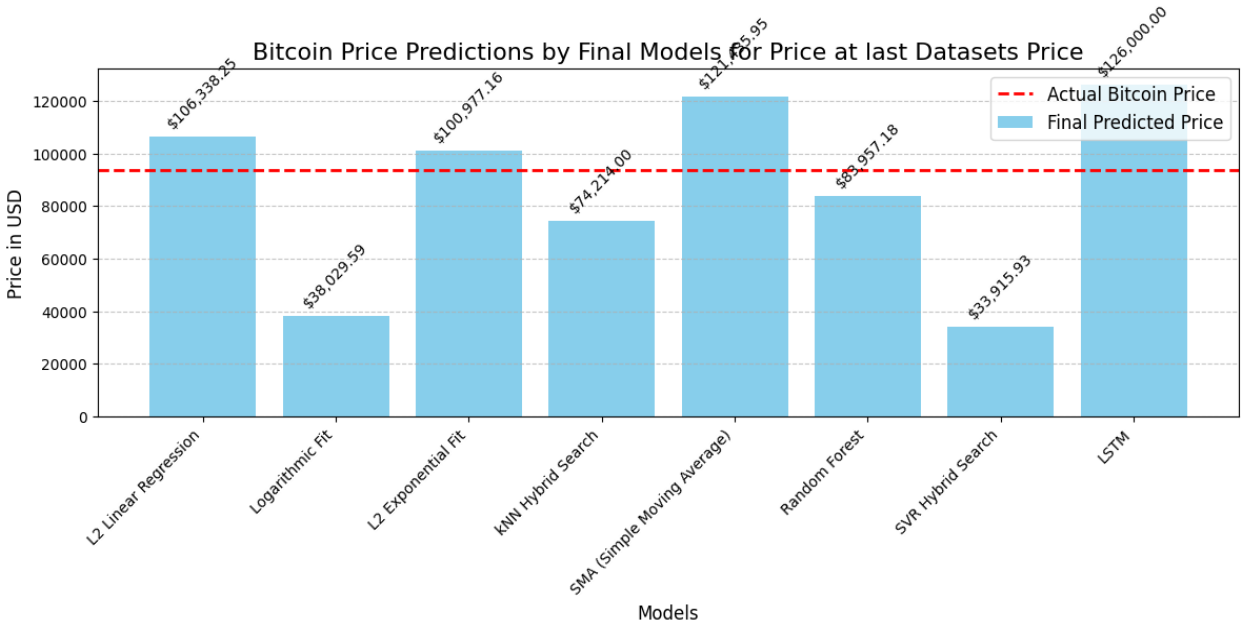
SMA = 121435.95

Random Forest = 83957.18

SVR(Hybrid-or Grid) = 33915.93

LSTM = 126000.0

10.Σύγκριση Μοντέλων(Καλύτερων)



Επομένως Καταλαβαίνουμε πως με βάση τα αρχικά μας μοντέλα, όλες τις συγκεκριμένες (υπερ)παραμέτρους και τις συνεχείς βελτιστοποιήσεις, το Καλύτερο Μοντέλο είναι το Exponential Fit!

Παρατηρούμε λοιπόν ότι παρόλλο την μεγαλη μεταβλητότητα του BTC , το καλύτερο μοντέλο ήταν ένα «Απλό». – Με τον ίδιο τρόπο έχει περίπου υπολογιστεί το Rainbow chart το οποίο έχει μέχρι στιγμής 100% ακρίβεια στην πρόβλεψη τιμών Αγοράς και Πώλησης BTC!!

- ***https://www.blockchaincenter.net/en/bitcoin-rainbow-chart/***

Προφανώς εάν θέλαμε να έχουμε μεγαλύτερη ακρίβεια θα μπορούσαμε να 'πειράξουμε' κι άλλο τις υπερπαραμέτρους και τις μετρικές μας ή να επικεντρωνόμασταν στα καλύτερα διαγράμματα για εύρεση ακόμη καλύτερων τιμών!!