ACCURACY

DAMAN VIRDI

HOW TO EVALUATE YOUR MACHINE LEARNING MODELS WITH PYTHON CODE!

You've finally built your machine learning model to predict future prices of Bitcoin so that you can finally become a multi-billionaire. But how do you know that the model you created is any good?

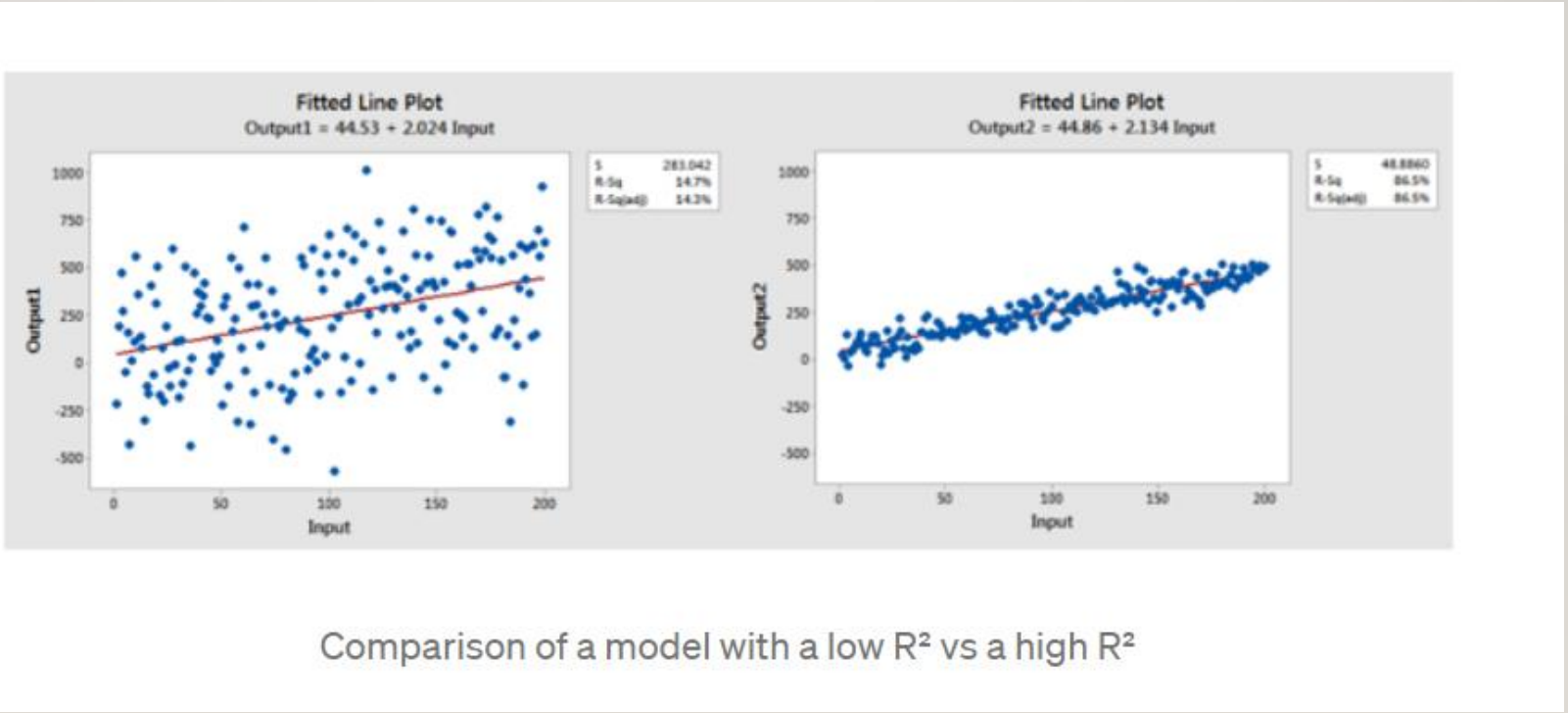
EVALUATING REGRESSION MODELS

1. R-Squared
2. Adjusted R-Squared
3. Mean Absolute Error
4. Mean Squared Error

R SQUARED

- R Squared is a measurement that tells you to what extent the proportion of variance in the dependent variable is explained by the variance in the independent variables. In simpler terms, while the coefficients estimate trends, R-squared represents the scatter around the line of best fit.
- For example, if the R^2 is 0.80, then 80% of the variation can be explained by the model's inputs.
- If the R^2 is 1.0 or 100%, that means that all movements of the dependent variable can be entirely explained by the movements of the independent variables.

TO SHOW A VISUAL EXAMPLE, DESPITE HAVING THE SAME LINE OF BEST FIT, THE R^2 ON THE RIGHT IS MUCH HIGHER THAN THE ONE ON THE LEFT



Comparison of a model with a low R^2 vs a high R^2

- The equation for R^2 is as follows

$$R^2 = 1 - \frac{Explained\ Variation}{Total\ Variation}$$

The Explained Variation is equal to the sum of squared residuals while the total variation is equal to the total sum of squared

$$SS_{residual} = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2$$

$$SS_{total} = \sum_{i=0}^{n} (y_i - y_{avg})^2$$

CODE SNIPPET FOR R-SQUARED

```
from sklearn.metrics import r2_score
sklearn.metrics.r2_score(y_true, y_pred)
```

2. ADJUSTED R-SQUARED

- Every additional independent variable added to a model **always** increases the R^2 value — therefore, a model with several independent variables may seem to be a better fit even if it isn't.
- This is where Adjusted R^2 comes in.
- The adjusted R^2 compensates for each additional independent variable and only increases if each given variable improves the model above what is possible by probability.

Option 1: Manual Calculation

```
# n = number of sample size
# p = number of independent variables

Adj_r2 = 1-(1-R2)*(n-1)/(n-p-1)
```

Option 2: statsmodel.api

```
import statsmodels.api as sm
from statsmodels.sandbox.regression.predstd import wls_prediction_std

model1=sm.OLS(y_train,x_train)
result=model1.fit()
print(result.summary())
```

3 . MEAN ABSOLUTE ERROR (MAE)

- The absolute error is the difference between the predicted values and the actual values. Thus, the mean absolute error is the average of the absolute error.

$$\text{Mean Absolute Error} = \frac{1}{N}\sum_{i=0}^{n} |y_i - \hat{y}_i|$$

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_true, y_pred)
```

4. MEAN SQUARED ERROR (MSE)

- The mean squared error or MSE is similar to the MAE, except you take the average of the **squared differences** between the predicted values and the actual values.
- Because the differences are squared, larger errors are weighted more highly, and so this should be used over the MAE when you want to minimize large errors. Below is th

$$\text{Mean Squared Error} = \frac{1}{N}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2$$

```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_true, y_pred)
```

B) EVALUATING CLASSIFICATION MODELS

1. Confusion Matrix and related metrics
2. F1 Score
3. AUC-ROC Curve

1 . CONFUSION MATRIX AND RELATED METRICS

- A confusion matrix, also known as an error matrix, is a performance measurement for assessing classification models. Below is an example of a two-class confusion matrix

	Predicted: No	Predicted: Yes
Actual: No	True Negative 50	False Positive 10
Actual: Yes	False Negative 5	True Positive 100

TERMINOLOGY

- **True Positive**: Outcome where the model correctly predicts the positive class.
- **True Negative**: Outcome where the model correctly predicts the negative class.
- **False Positive (Type 1 Error)**: Outcome where the model incorrectly predicts the positive class.
- **False Negative (Type 2 Error)**: Outcome where the model incorrectly predicts the negative class.

- **Accuracy:** equal to the fraction of predictions that a model got right.

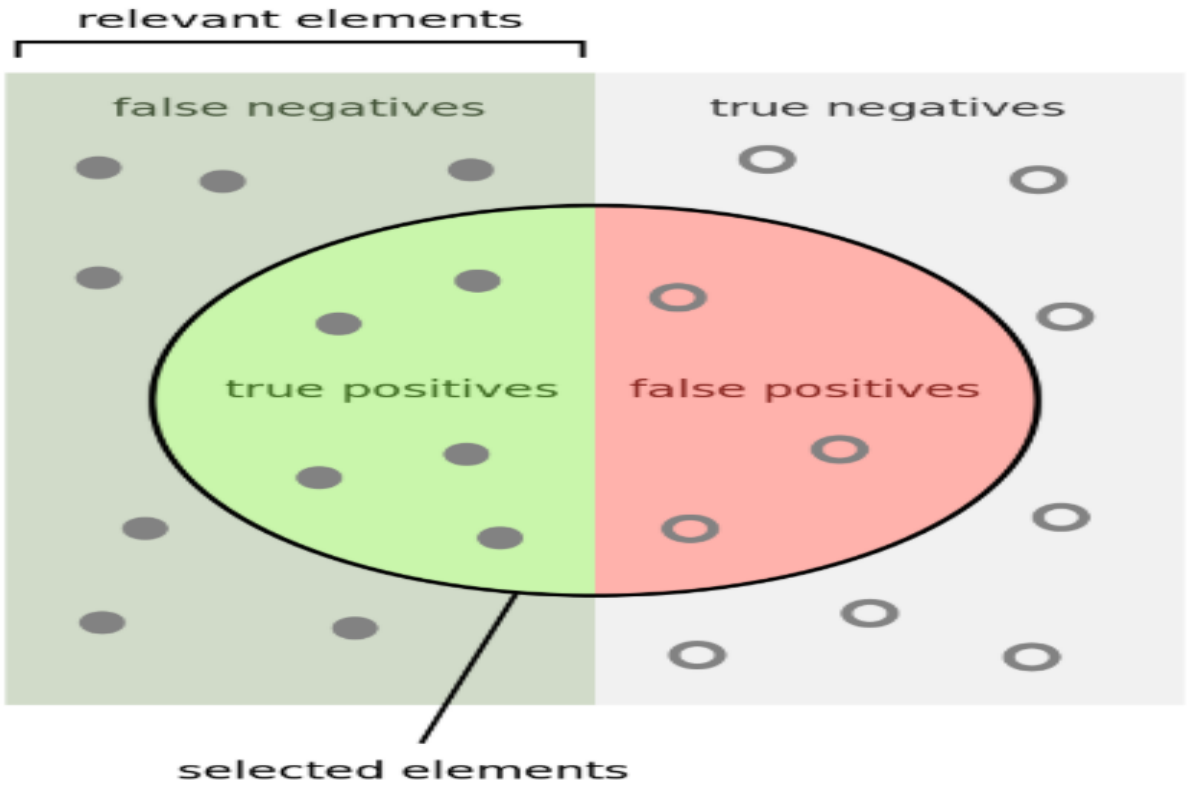
$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

- **Recall:** attempts to answer “What proportion of actual positives was identified correctly?”

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **Precision:** attempts to answer “What proportion of positive identifications was actually correct?”

$$\text{Precision} = \frac{TP}{TP + FP}$$



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{selected elements}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{relevant elements}}$$

Taken from Wikipedia

```
# Confusion Matrix
from sklearn.metrics import confusion_matrix
confusion_matrix(y_true, y_pred)


# Accuracy
from sklearn.metrics import accuracy_score
accuracy_score(y_true, y_pred)


# Recall
from sklearn.metrics import recall_score
recall_score(y_true, y_pred, average=None)


# Precision
from sklearn.metrics import precision_score
precision_score(y_true, y_pred, average=None)
```

2. F1 SCORE

- The F1 score is a measure of a test's accuracy — it is the harmonic mean of precision and recall. It can have a maximum score of 1 (perfect precision and recall) and a minimum of 0. Overall, it is a measure of the preciseness and robustness of your model.

$$F1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{2 * (precision * recall)}{precision + recall}$$

THERE ARE THREE WAYS YOU CAN CALCULATE THE F1 SCORE IN PYTHON:

```
# Method 1: sklearn
from sklearn.metrics import f1_score
f1_score(y_true, y_pred, average=None)


# Method 2: Manual Calculation
F1 = 2 * (precision * recall) / (precision + recall)


# Method 3: BONUS - classification report
from sklearn.metrics import classification_report
print(classification_report(y_true, y_pred,
target_names=target_names))
```

3. AUC-ROC CURVE

- The AUC-ROC Curve is a performance measurement for classification problems that tells us how much a model is capable of distinguishing between classes. A higher AUC means that a model is more accurate.

```
import numpy as np
from sklearn.metrics import roc_auc_score


y_true = np.array([0, 0, 1, 1])
y_scores = np.array([0.1, 0.4, 0.35, 0.8])
roc_auc_score(y_true, y_scores)
0.75
```