INTRODUCTION TO LINEAR MODELING IN PYTHON

Modeling Real Data

Jason Vestuto
Data Scientist

Scikit-Learn

```
from sklearn.linear_model import LinearRegression
```

```
# Initialize a general model
model = LinearRegression(fit_intercept=True)
```

```
# Load and shape the data
x_raw, y_raw = load_data()
x_data = x_raw.reshape(len(y_raw),1)
y_data = y_raw.reshape(len(y_raw),1)
```

```
# Fit the model to the data
model_fit = model.fit(x_data, y_data)
```

Predictions and Parameters

```
# Extract the linear model parameters
intercept = model.intercept_[0]
slope = model.coef_[0,0]
```

```
# Use the model to make predictions
future_x = 2100
future_y = model.predict(future_x)
```

statsmodels

```
x, y = load_data()
df = pd.DataFrame(dict(times=x_data, distances=y_data))
```

```
fig = df.plot('times', 'distances')
```

```
model_fit = ols(formula="distances ~ times", data=df).fit()
```

Uncertainty

```
a0 = model_fit.params['Intercept']
a1 = model_fit.params['times']
```

```
e0 = model_fit.bse['Intercept']
e1 = model_fit.bse['times']
```

```
intercept = a0
slope = a1
uncertainty_in_intercept = e0
uncertainty_in_slope = e1
```

INTRODUCTION TO LINEAR MODELING IN PYTHON

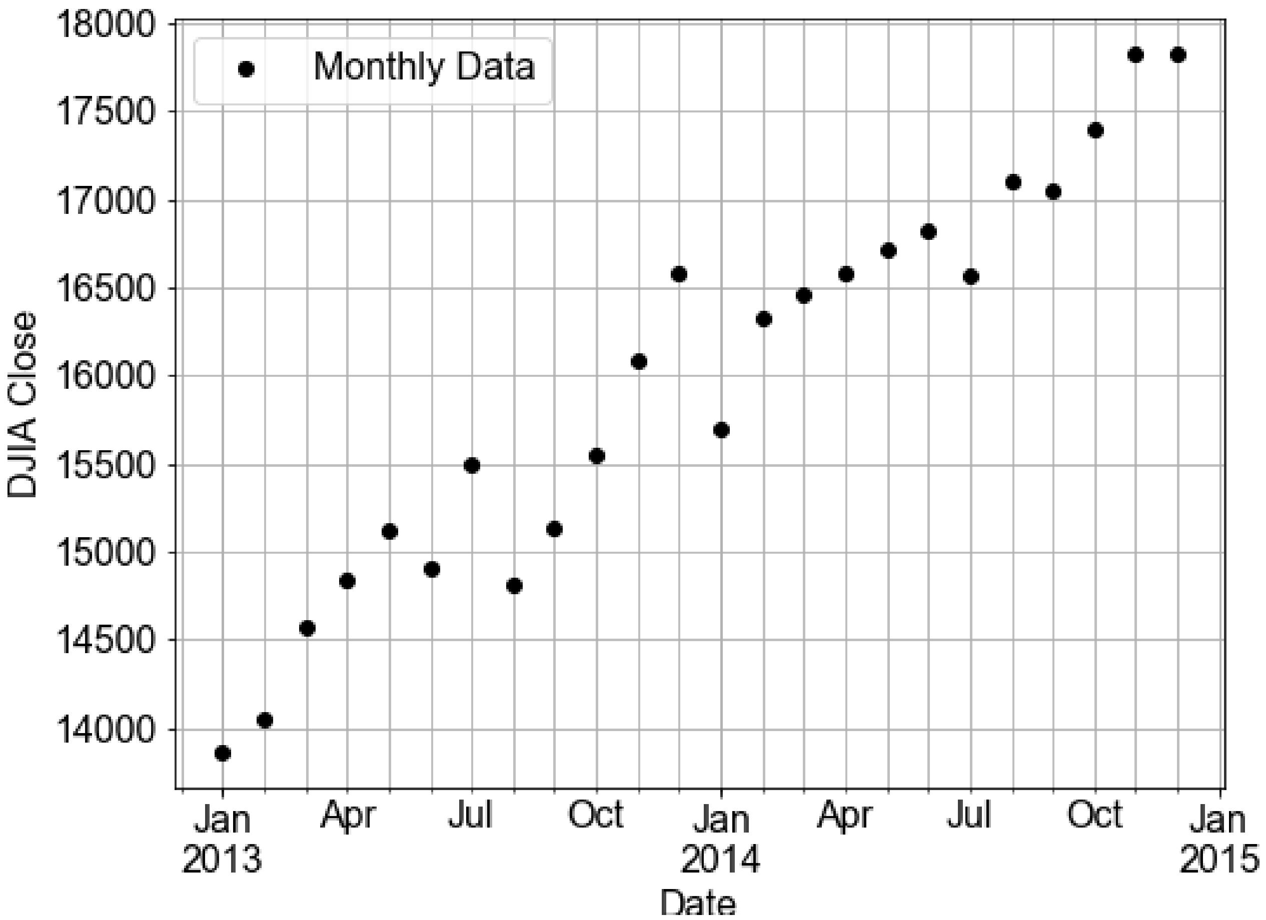
Let's practice!

INTRODUCTION TO LINEAR MODELING IN PYTHON

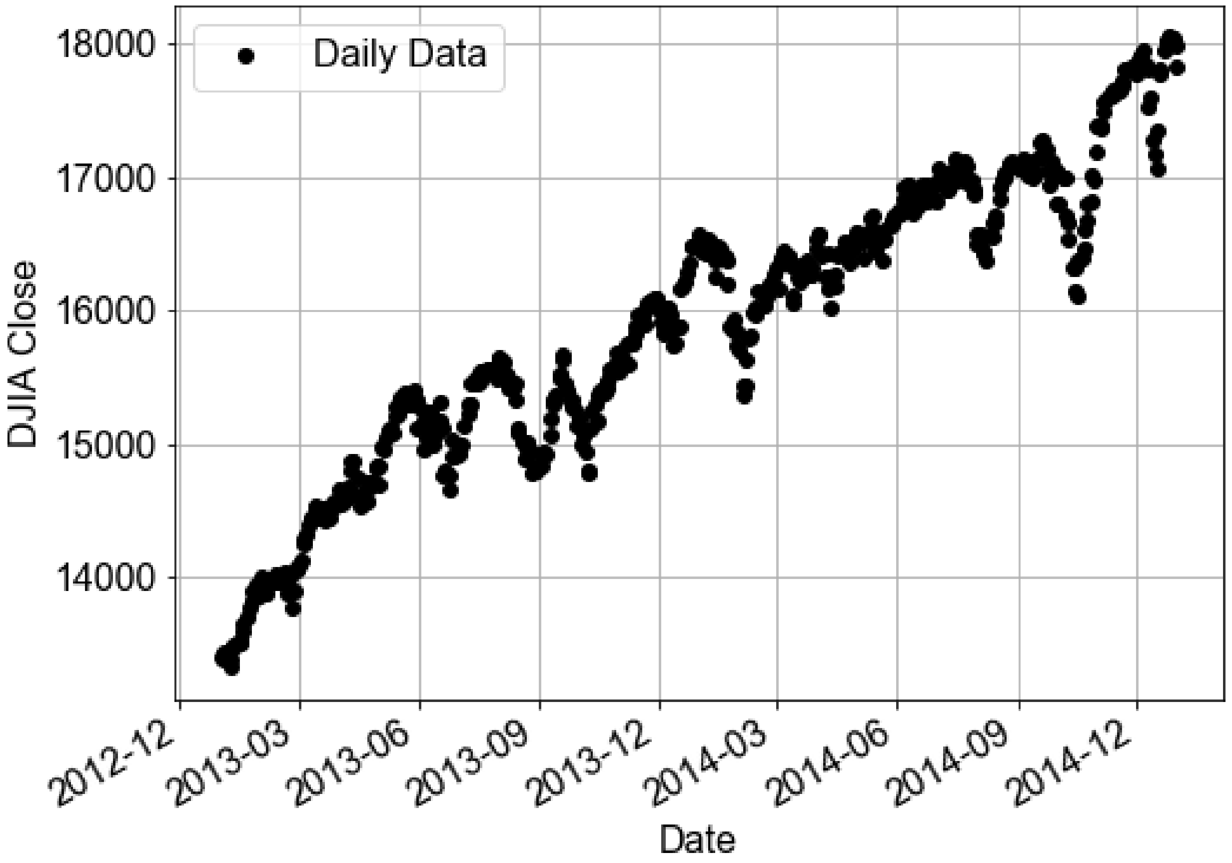
The Limits of Prediction

Jason Vestuto
Data Scientist

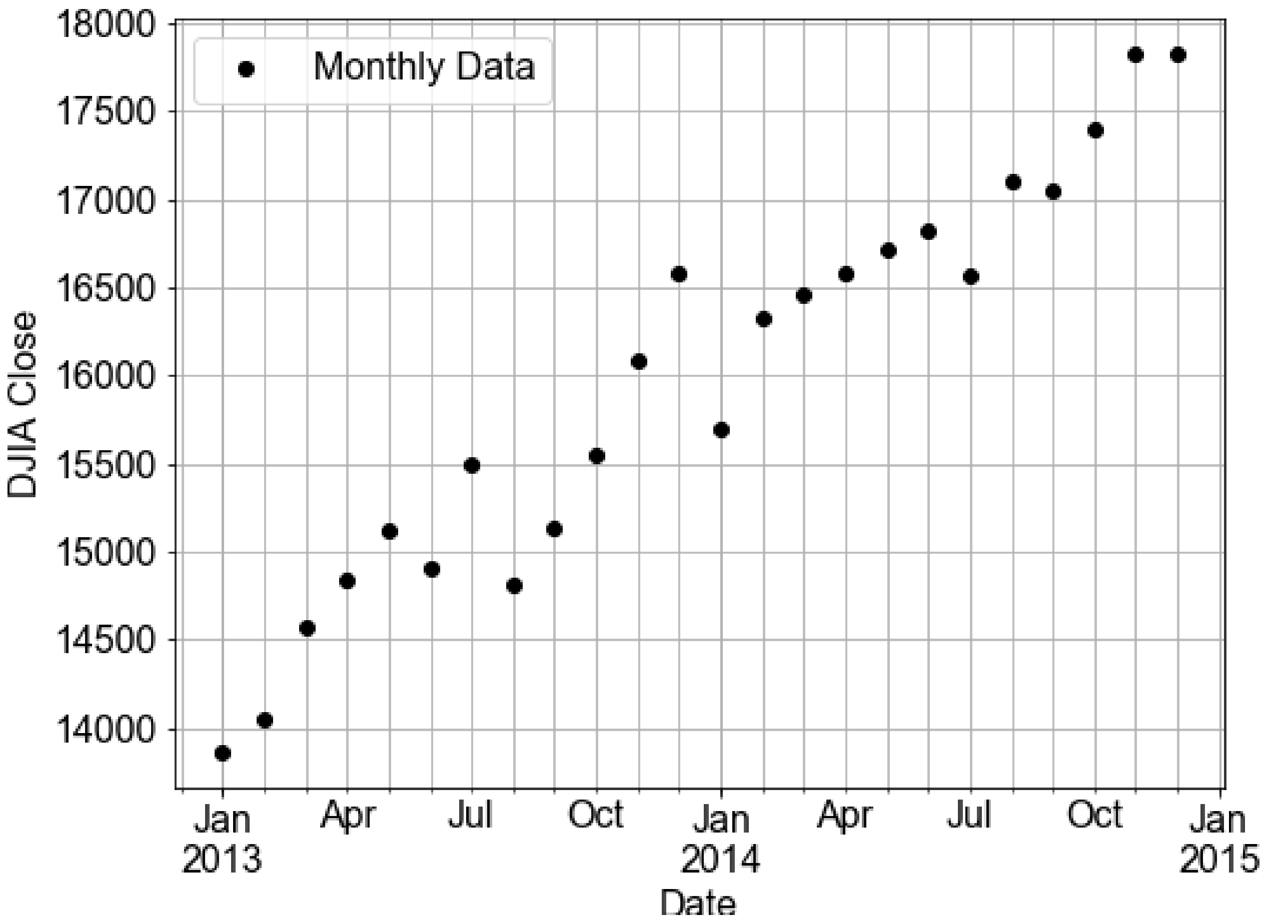
Interpolation



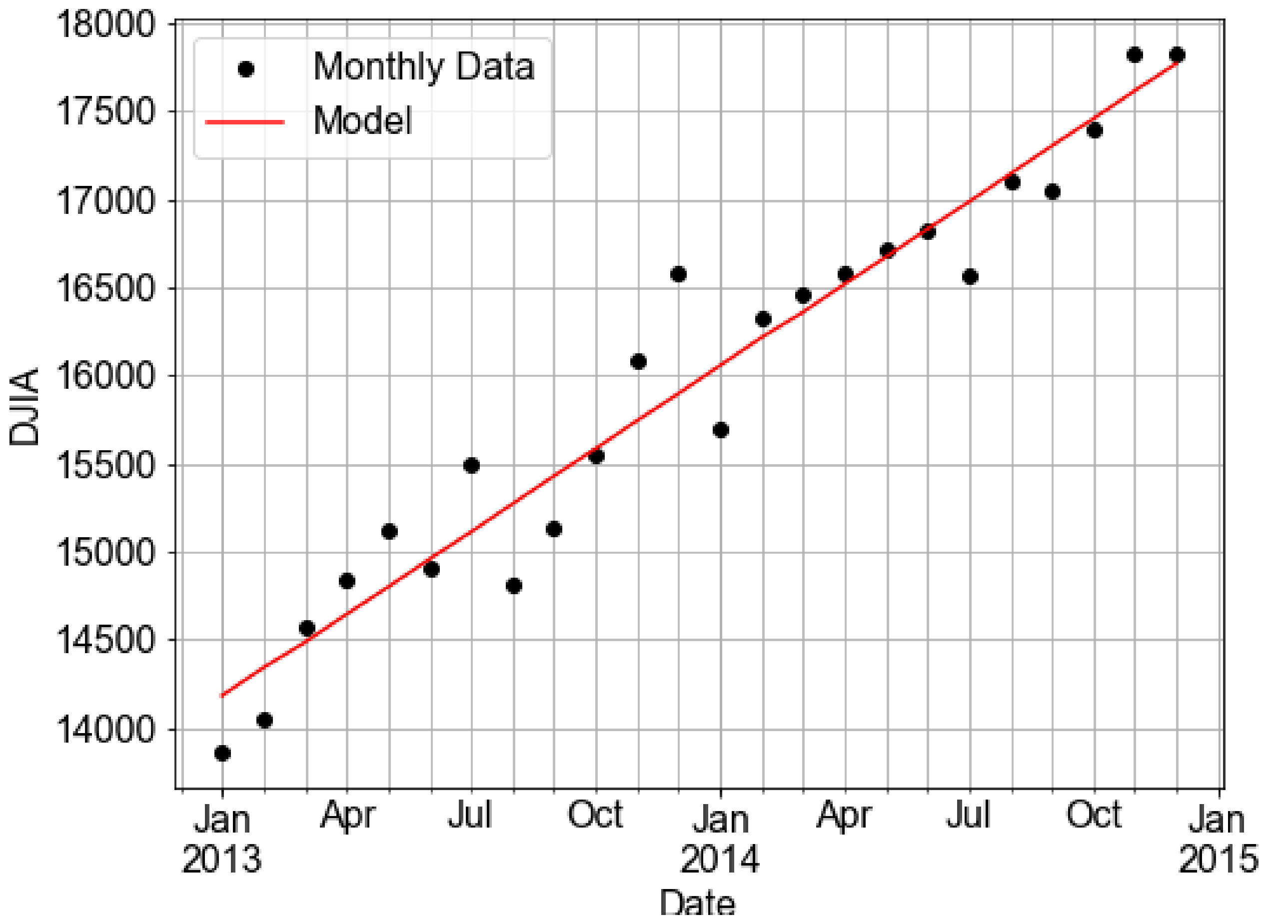
Interpolation



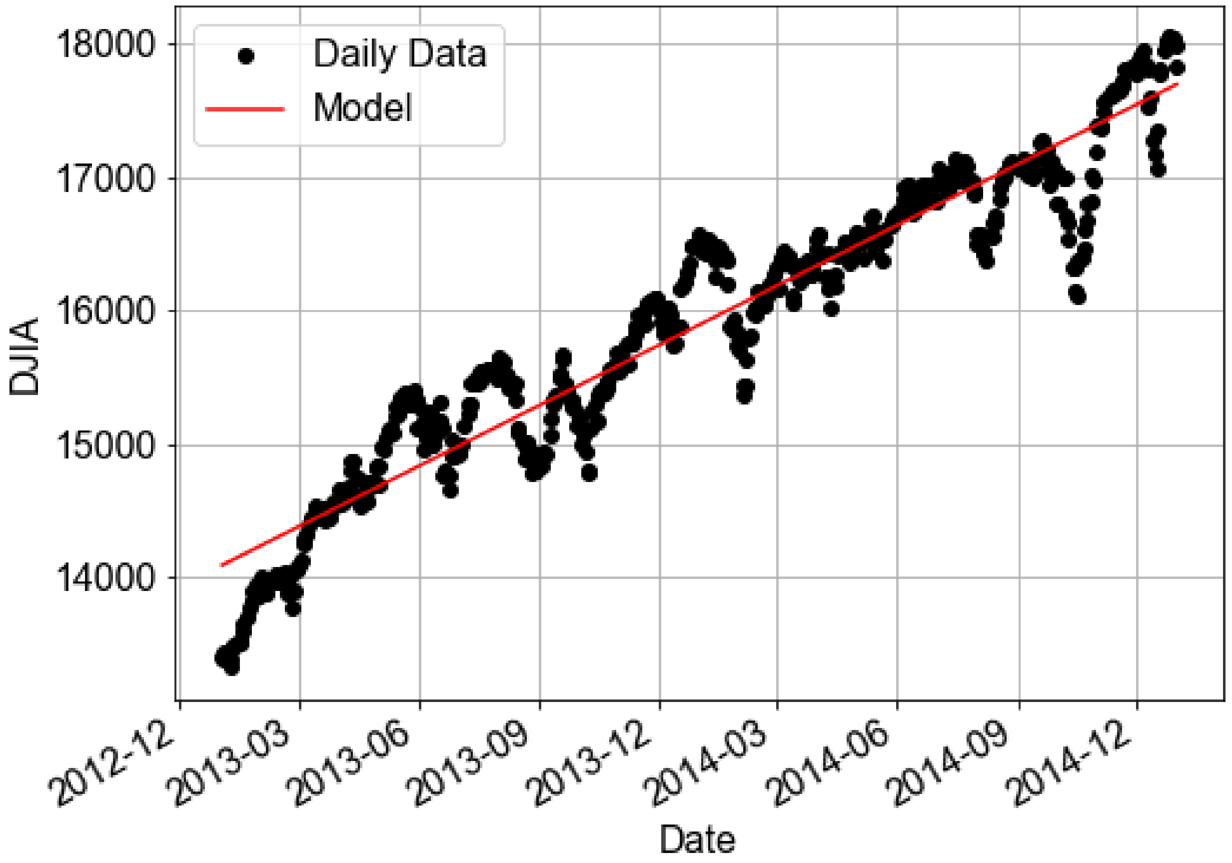
Interpolation



Interpolation

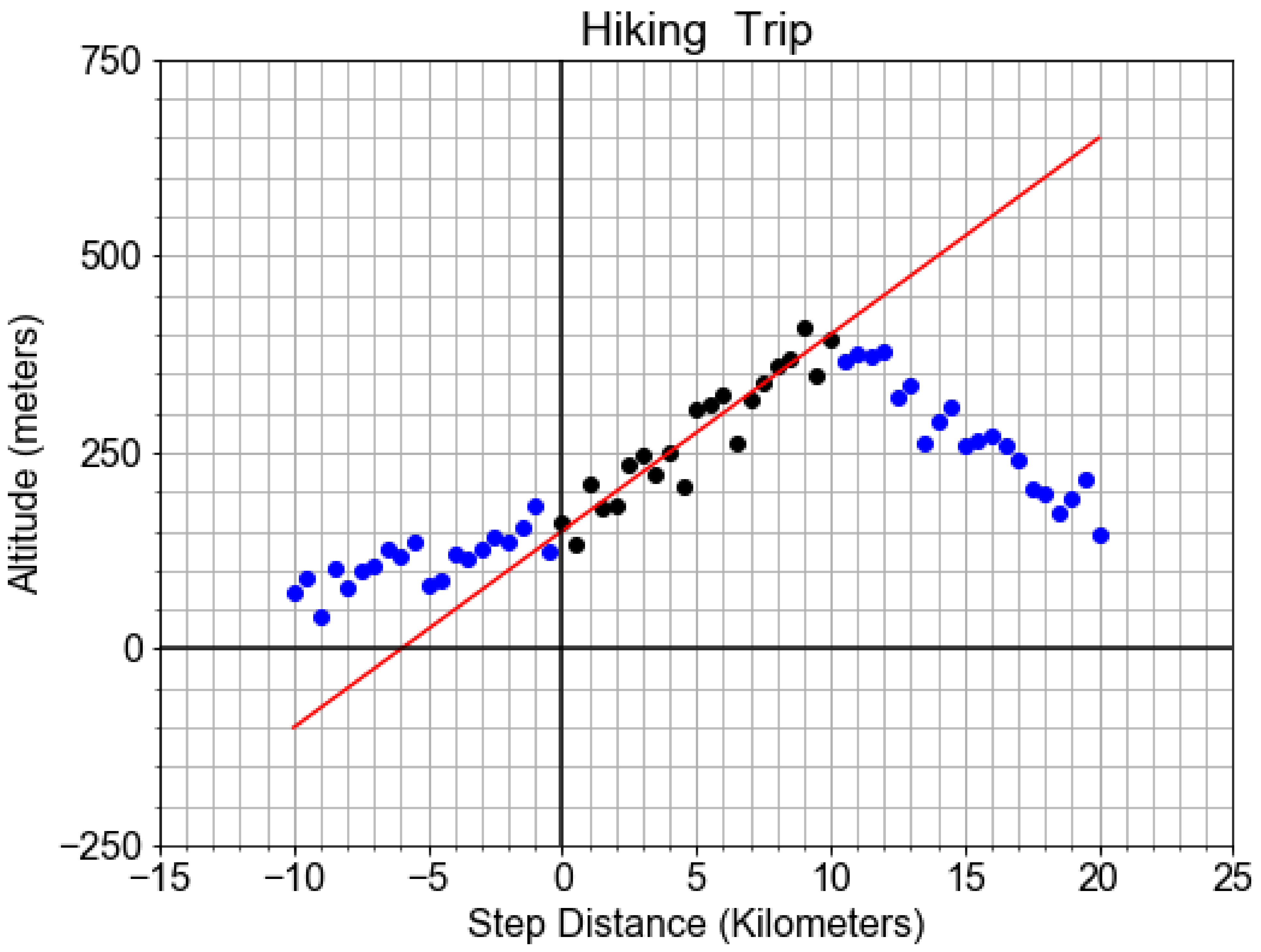


Interpolation



Domain of Validity

- zoom in: data looks linear
- model assumption: `a2*x**2 + a3*x**3` + ... = zero.
- build a linear model: `a0 + a1*x`
- zoom out: your model breaks


Hiking Trip
Altitude (meters)
750
500
250
0
−250
−15
−10
−5
0
5
10
15
20
25
Step Distance (Kilometers)

INTRODUCTION TO LINEAR MODELING IN PYTHON

Let's practice!

INTRODUCTION TO LINEAR MODELING IN PYTHON

Goodness-of-Fit

Jason Vestuto
Data Scientist

3 Different R's

Building Models:

- RSS

Evaluating Models:

- RMSE
- R-squared

RMSE

```
residuals = y_model - y_data
RSS = np.sum( np.square(residuals) )
```

```
mean_squared_residuals = np.sum( np.square(residuals) ) / len(residuals)
```

```
MSE = np.mean( np.square(residuals) )
```

```
RMSE = np.sqrt(np.mean( np.square(residuals)))
```

```
RMSE = np.std(residuals)
```

R-Squared in Code

Deviations:

```
deviations = np.mean(y_data) - y_data
VAR = np.sum(np.square(deviations))
```

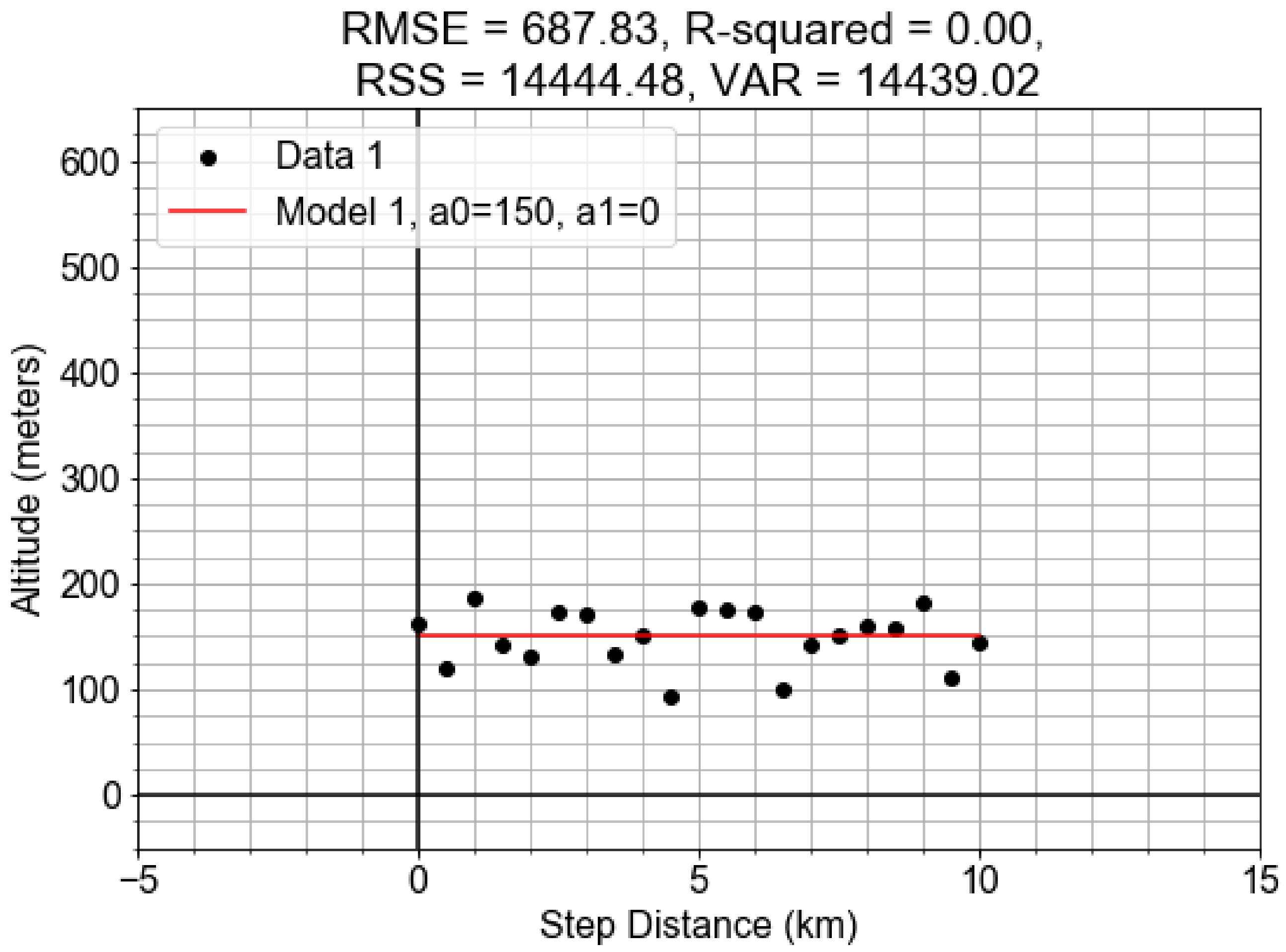
Residuals:

```
residuals = y_model - y_data
RSS = np.sum(np.square(residuals))
```

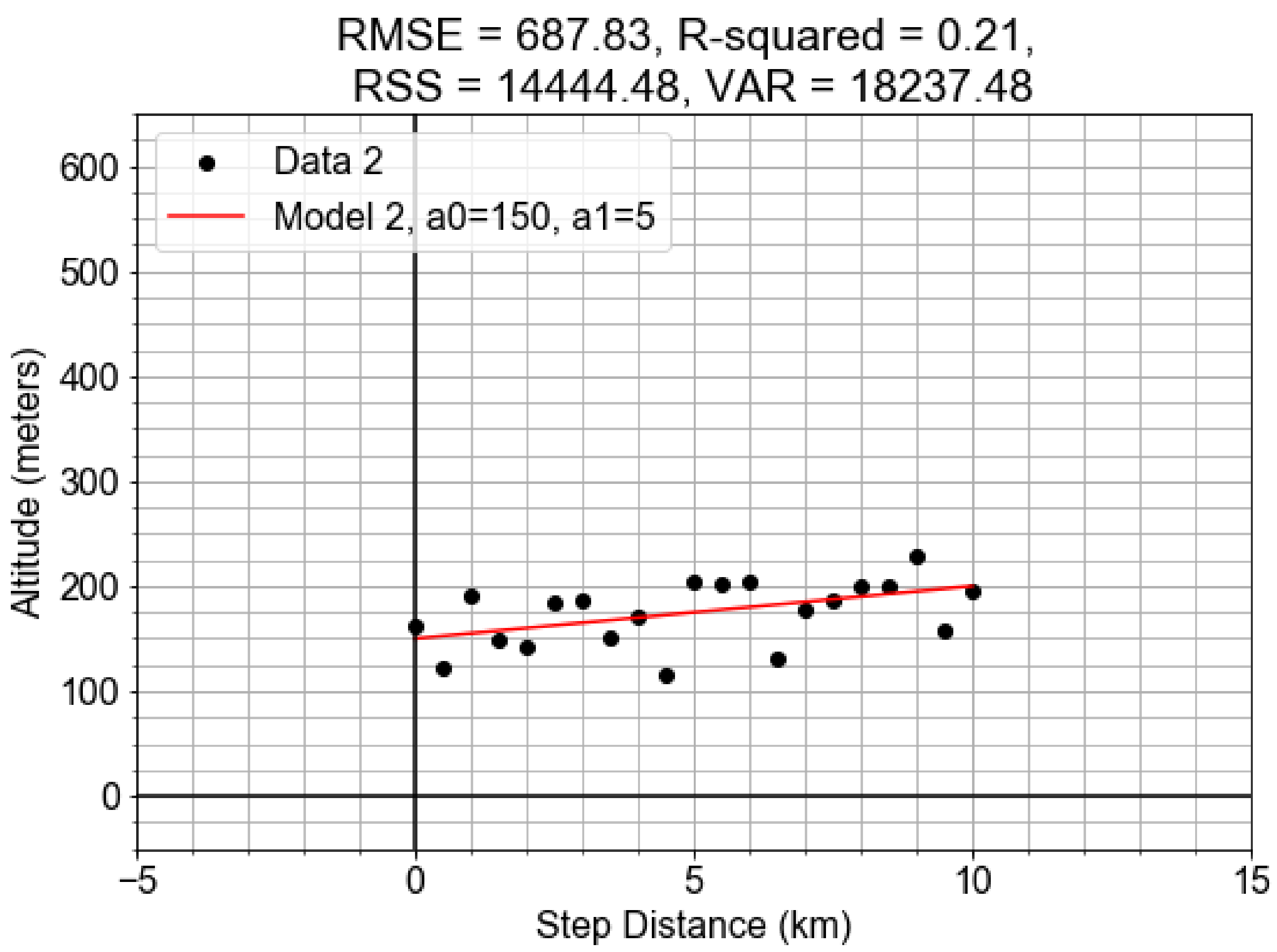
R-squared:

```
r_squared = 1 - (RSS / VAR)
r = correlation(y_data, y_model)
```

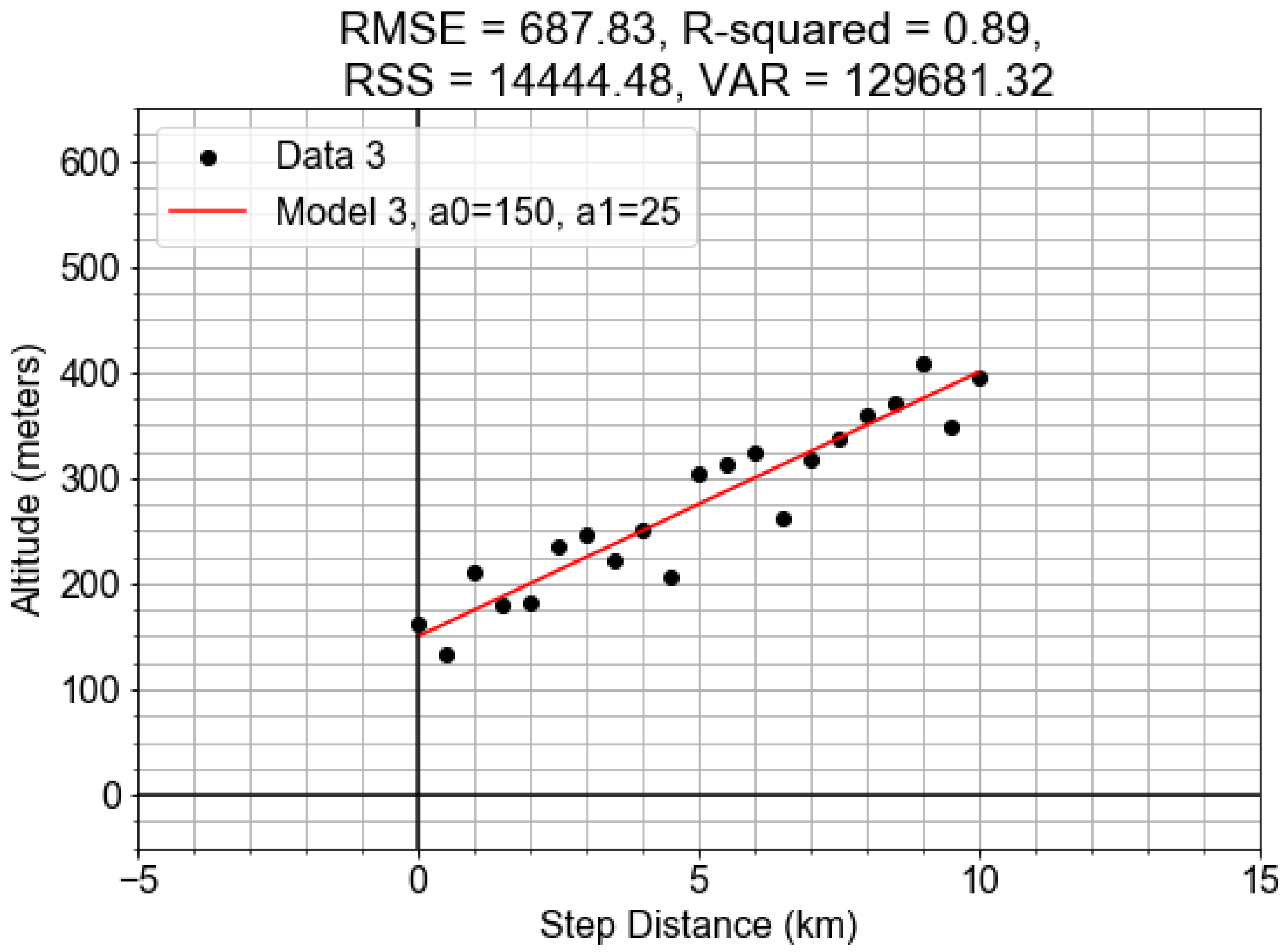
R-Squared in Data



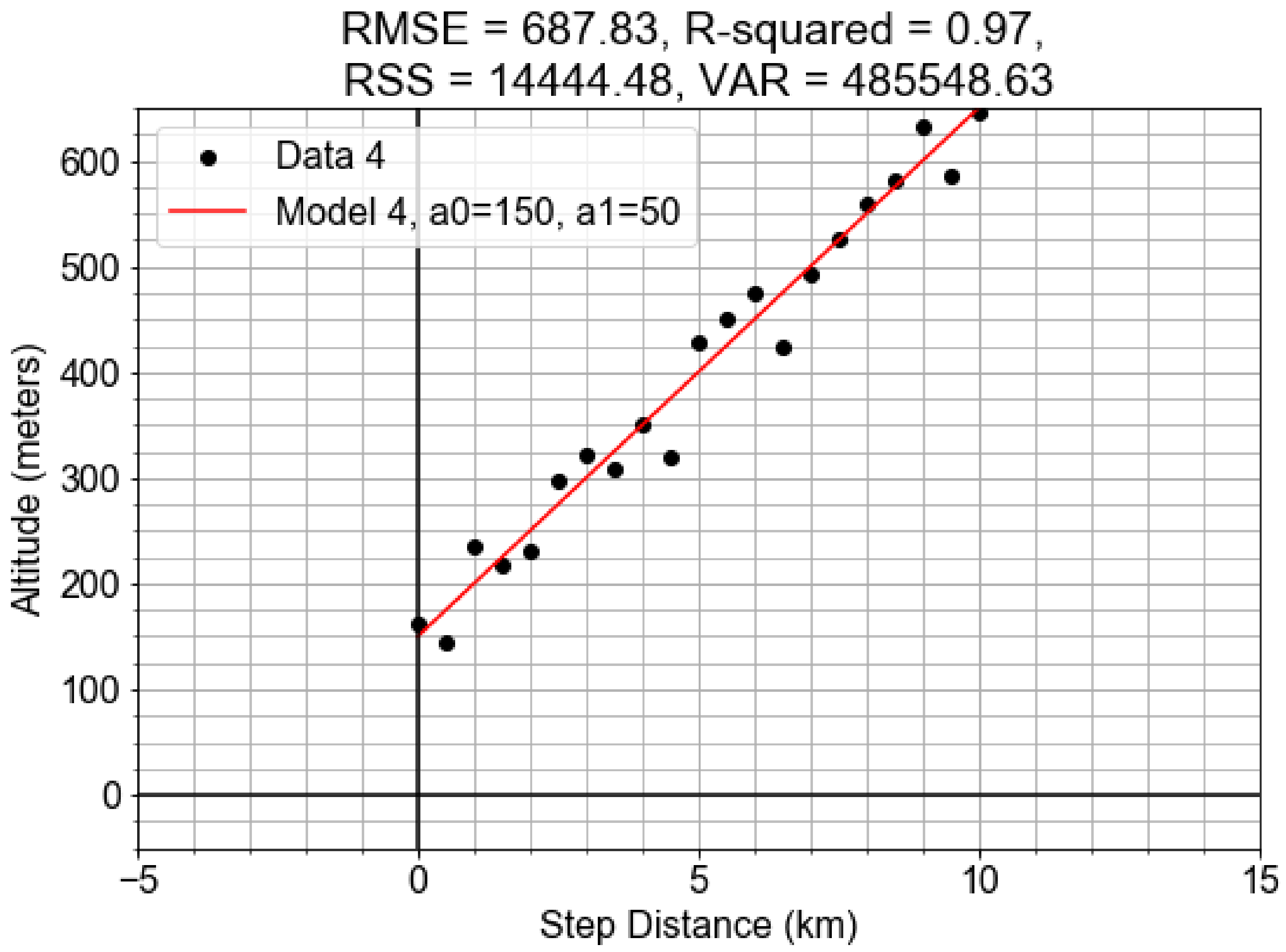
R-Squared in Data



R-Squared in Data



R-Squared in Data



RMSE vs R-Squared

- RMSE: how much variation is residual
- R-squared: what fraction of variation is linear

INTRODUCTION TO LINEAR MODELING IN PYTHON

Let's practice!

INTRODUCTION TO LINEAR MODELING IN PYTHON

Standard Error

Jason Vestuto
Data Scientist

Uncertainty in Predictions

Model Predictions and RMSE:

- predictions compared to data gives residuals
- residuals have spread
- RMSE, measures residual spread
- RMSE, quantifies prediction goodness

Uncertainty in Parameters

Model Parameters and Standard Error:

- Parameter value as center
- Parameter standard error as spread
- Standard Error, measures parameter uncertainty

Computing Standard Errors

```
df = pd.DataFrame(dict(times=x_data, distances=y_data))
```

```
model_fit = ols(formula="distances ~ times", data=df).fit()
```

```
a1 = model_fit.params['times']
a0 = model_fit.params['Intercept']
```

```
slope = a1
intercept = a0
```

```
e0 = model_fit.bse['Intercept']
e1 = model_fit.bse['times']
```

```
standard_error_of_intercept = e0
standard_error_of_slope = e1
```

INTRODUCTION TO LINEAR MODELING IN PYTHON

Let's practice!