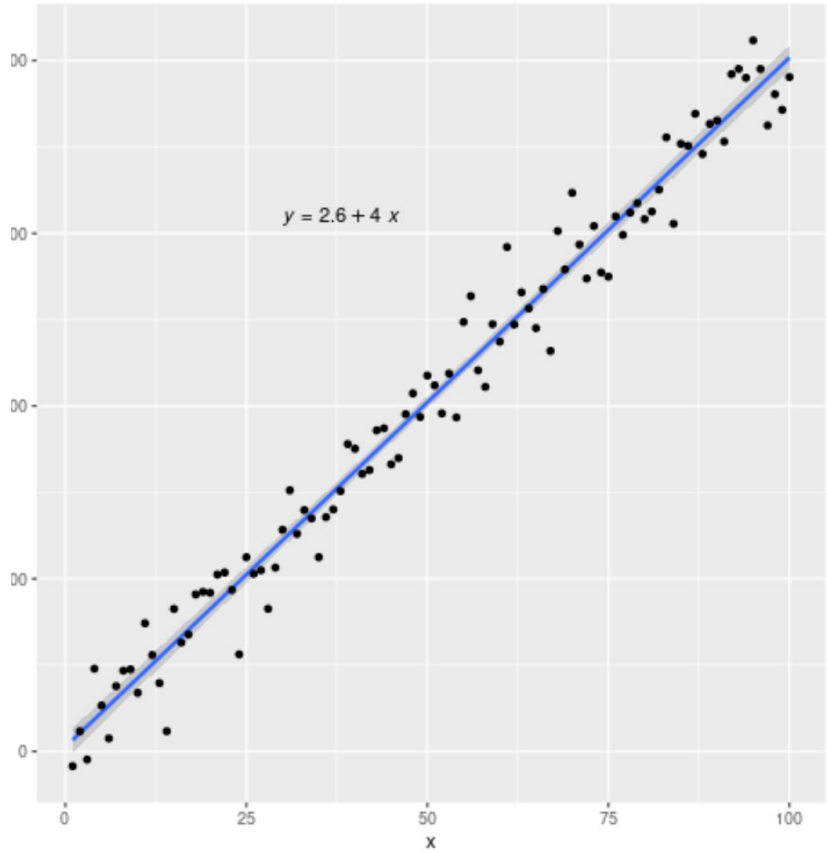
Statistics in Data Science

overview

1. Statistical Hypothesis Testing
2. Chi-Squared Goodness-Of-Fit Test
3. Linear Regression

Introduction to Linear Regression

- A Fundamental Machine Learning Algorithm
- Image: Linear regression equation and scatter plot
- Why are we learning linear regression?
    1. Widely used
    2. Runs fast
    3. Easy to implement (not a lot of tuning required)
    4. Highly interpretable
    5. Basis for many other methods



Linear Regression

- Definition: Linear regression is a supervised machine learning algorithm used for predicting a continuous target variable (dependent variable) based on one or more predictor variables (independent variables) by fitting a linear equation.

- Simple linear regression:

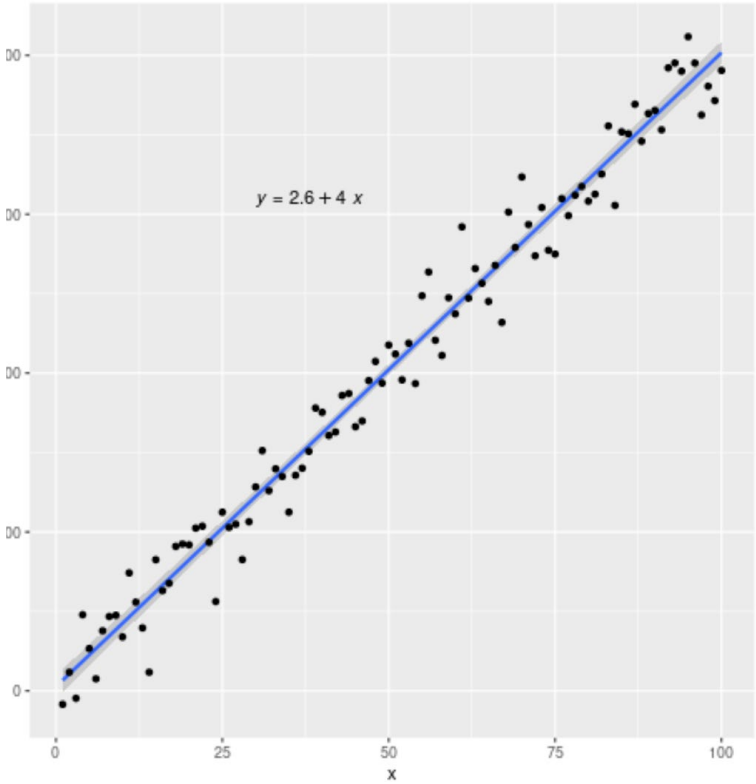
  Formula: Y = mx + b

- Multiple linear regression:

  Formula: Y = b0 + b1X1 + b2X2 + ... + bnXn

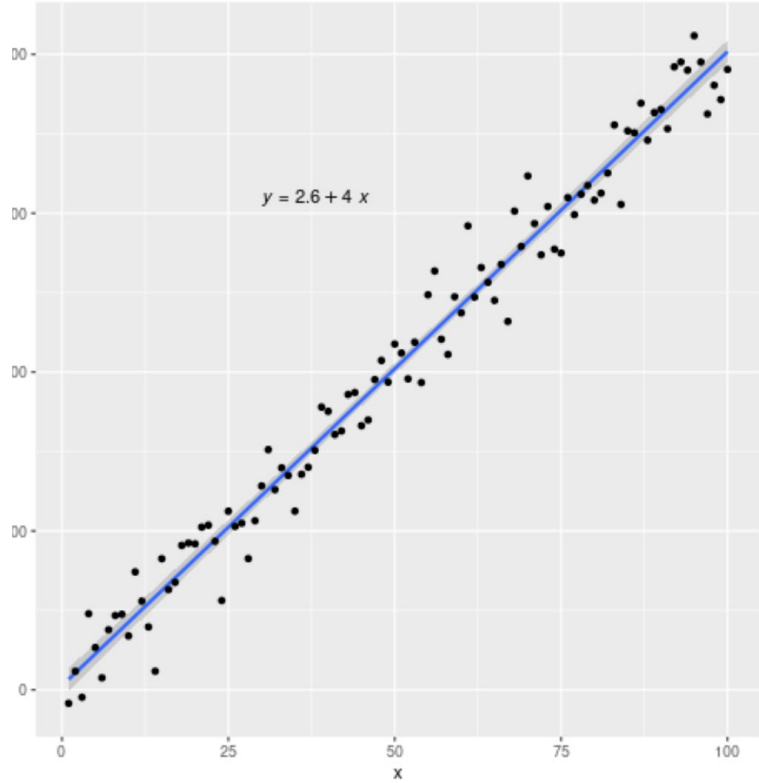
- Polynomial linear regression:

  Formula: Y = b0 + b1X1 + b2X1^2 + ... + bnX1^n



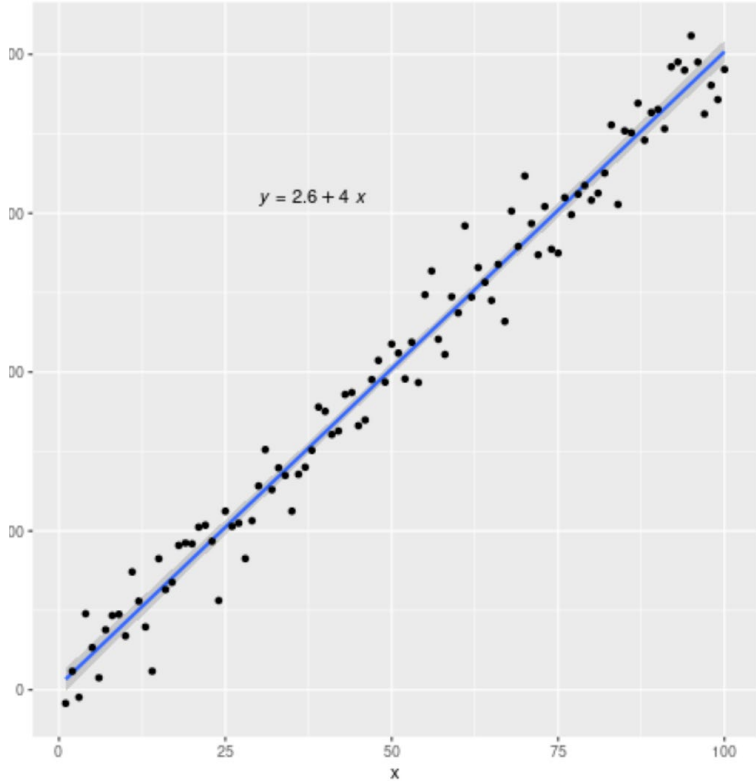
Key Concepts

- Dependent Variable (Y)
- Independent Variable (X)
- Slope (m) and Intercept (b) (Y = mx + b)
  - The slope represents the change of Y over X.
    It quantifies the direction and steepness
    of the relationship between X and Y.
  - The intercept represents the estimated value
    of Y when all independent variables are zero.



Key Concepts

- Residuals
  - The differences between the actual observed values of the dependent variable (Y) and the predicted values ($\hat{Y}$) generated by the linear regression model.
  - $e = Y - \hat{Y}$.
  - indicate how well the model fits the data.
- Line of Best Fit
  - The straight line of linear regression model.
  - Best fit the data points
  - Make predictions and understand the relationship between variables.



Simple Linear Regression

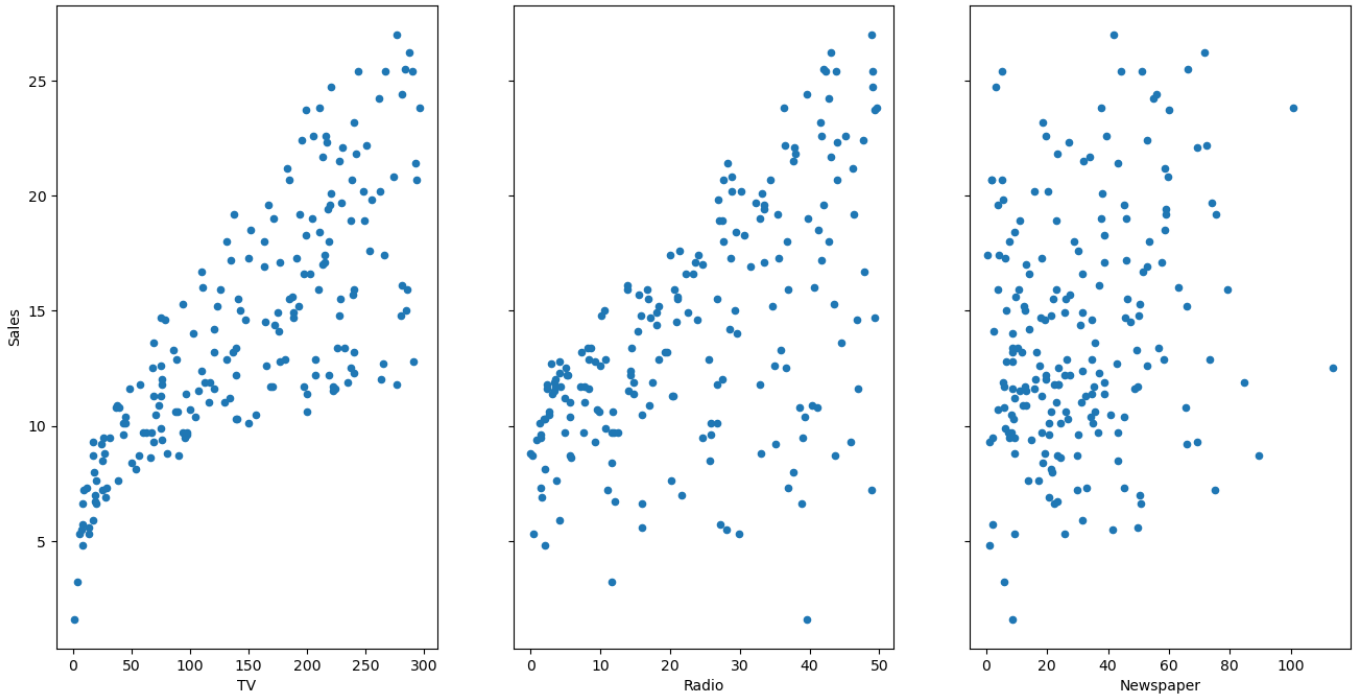
- Definition: Simple linear regression uses one independent variable to predict the dependent variable.
- Example: Predicting house prices based on the square footage of the house.
- Formula: $Y = mx + b$
- Term m, b are called the model coefficients.
  - Estimate model coefficients: least squares method

Example:

Features: TV, Radio, Newspaper
Response: Sales

```
# read data into a DataFrame
data = pd.read_csv('Data/Advertising.csv', index_col=0)
data.head()
```

	TV	Radio	Newspaper	Sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9

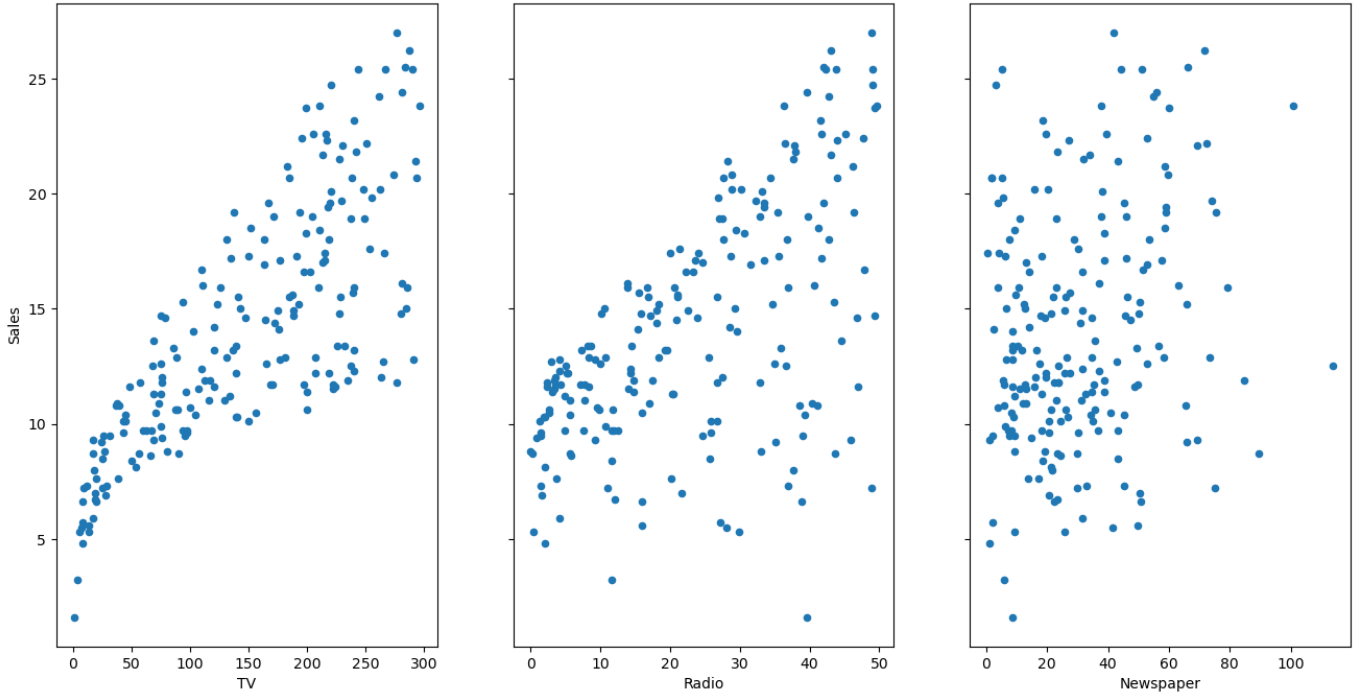


Example:

Features: TV, Radio, Newspaper
Response: Sales

Questions About the Advertising Data

1.Is there a relationship between ads and sales?
2.How strong is that relationship?
3.Which ad types contribute to sales?
4.What is the effect of each ad type of sales?
5.Given ad spending in a particular market, can sales be predicted?



Example:

Back to our Advertisement Example:

Let's use `statsmodels` to estimate the model coefficients for the advertising data:

```
# this is the standard import if you're using "formula notation" (similar to R)
import statsmodels.formula.api as smf

# create a fitted model in one line
lm = smf.ols(formula='Sales ~ TV', data=data).fit()

# print the coefficients
lm.params
```

```
Intercept    7.032594
TV           0.047537
dtype: float64
```

Using the Model for Prediction

Let's say that there was a new market where the TV advertising spend was **$50,000**. What would we predict for the Sales in that market?

$$y = \beta_0 + \beta_1 x$$

$$y = 7.032594 + 0.047537 \times 50$$

```
# manually calculate the prediction
7.032594 + 0.047537*50
```

```
9.409444
```

Thus, we would predict Sales of **9,409 widgets** in that market.

Of course, we can also use Statsmodels to make the prediction:

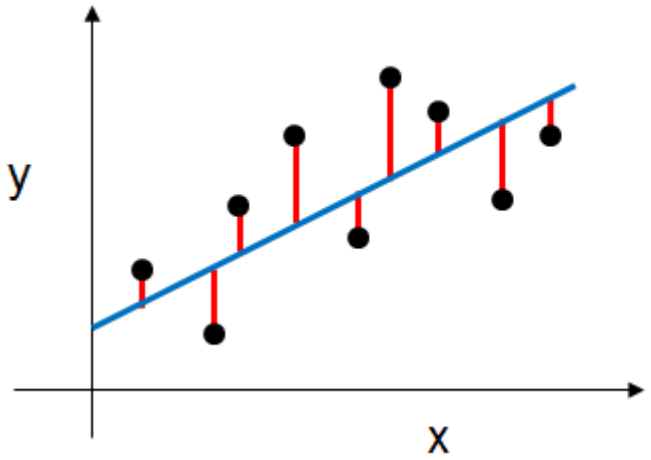
```
# you have to create a DataFrame since the Statsmodels formula interface expects it
X_new = pd.DataFrame({'TV': [50]})
X_new.head()
```

	TV
0	50

```
# use the model to make predictions on a new value
lm.predict(X_new)
```

```
0    9.409426
dtype: float64
```

Least Squares Method



$$SS_{residuals} = \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$$

Model Prediction

Observed Result

- black dots are the observed values of x and y.
- The blue line is our least squares line.
- The red lines are the residuals, distances between the observed values and the least squares line.
- By minimizing the sum of squared residuals (SSR), you are finding the values of m and b that provide the best fit for the linear relationship between the variables X and Y.

Example:

- **Plotting the Least Squares Line**
- Let's make predictions for the **smallest and largest observed values of x**, and then use the predicted values to plot the least squares line:

```
# create a DataFrame with the minimum and maximum values of TV
X_new = pd.DataFrame({'TV': [data.TV.min(), data.TV.max()]})
X_new.head()
```

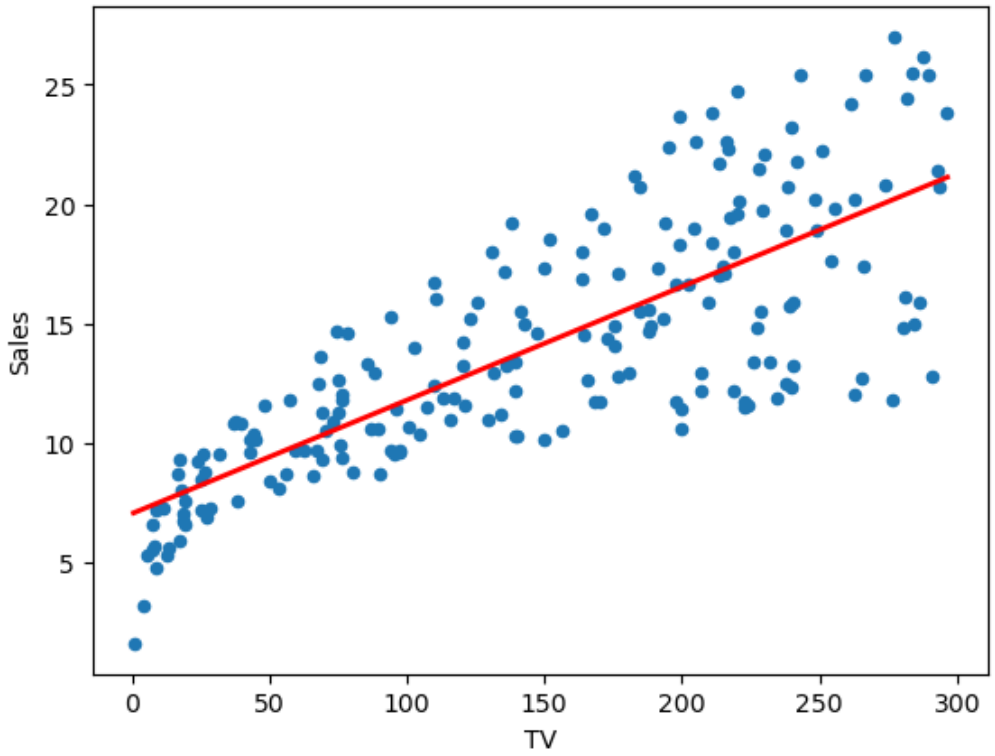
	TV
0	0.7
1	296.4

```
# make predictions for those x values and store them
preds = lm.predict(X_new)
preds
```

```
0     7.065869
1    21.122454
dtype: float64
```

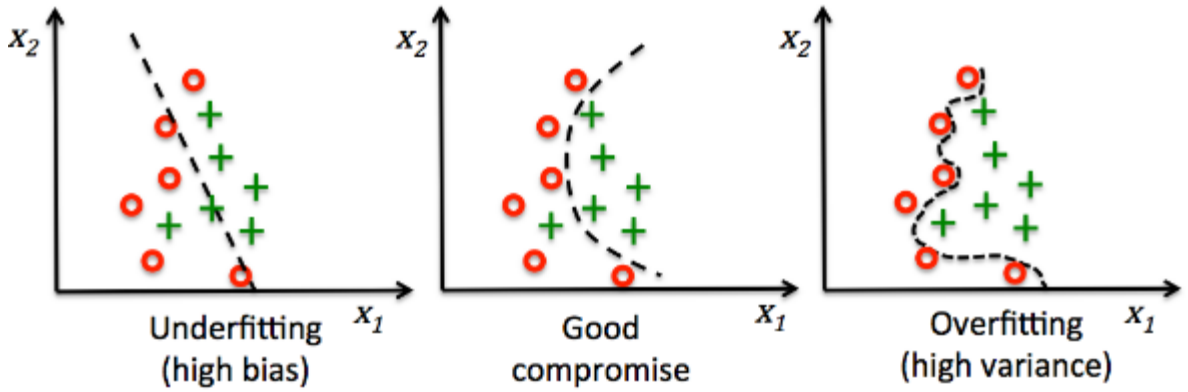
```
# first, plot the observed data
data.plot(kind='scatter', x='TV', y='Sales')

# then, plot the least squares line
plt.plot(X_new, preds, c='red', linewidth=2)
```



Bias vs Variance

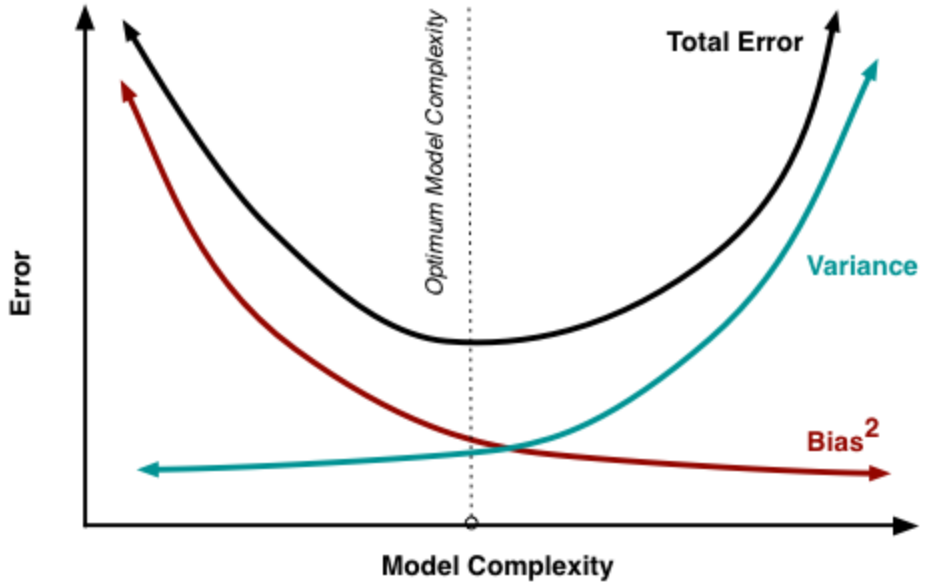
- Error introduced by linear regression
- Bias refers to the error introduced by making overly simplistic assumptions in the learning algorithm. It can lead the algorithm to consistently underpredict or overpredict the true values.
- Variance, on the other hand, refers to the error introduced by the model's sensitivity to small fluctuations or noise in the training data. It can lead the model to be overly complex and fit the training data too closely.



Bias vs Variance

- Bias is associated with errors from overly simplistic models, leading to underfitting and poor performance.
- Variance is associated with errors from overly complex models, leading to overfitting and poor generalization to new data.
- "bias-variance trade-off."

generalize well to unseen data while accurately capturing the underlying patterns in the training data



How Well Does the Model Fit the data?

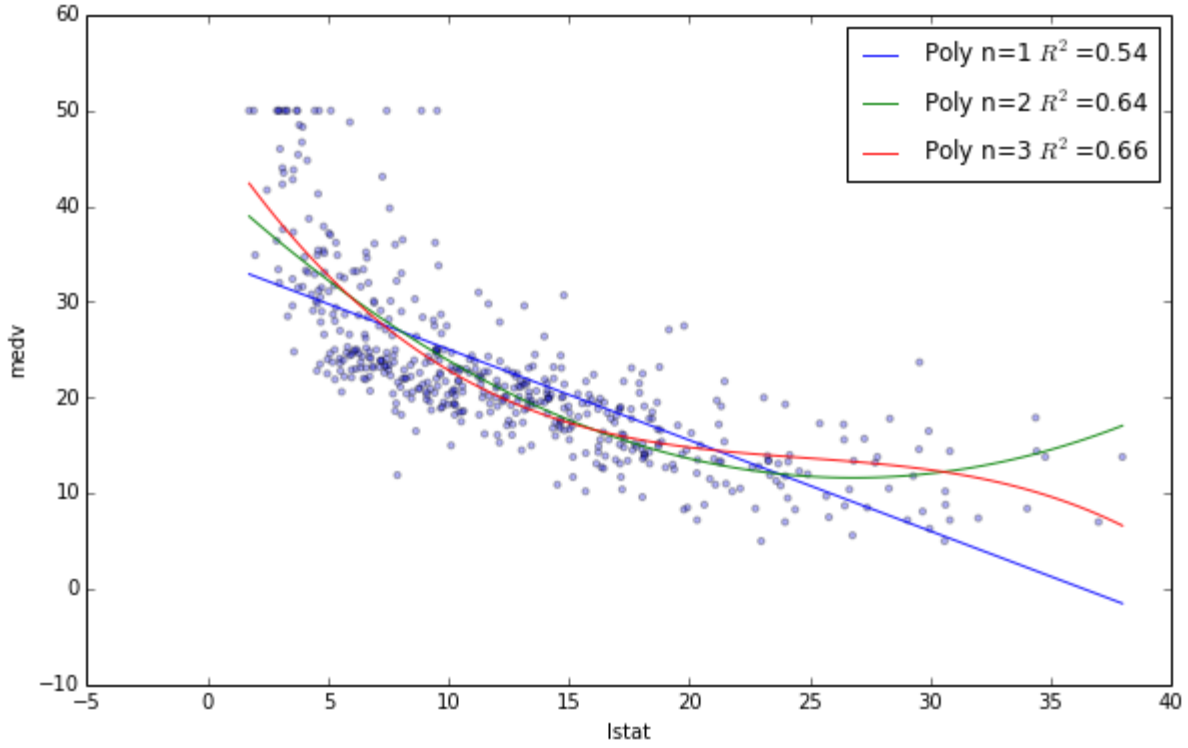
- R-squared (R^2) value is a statistical measure that quantifies how well a linear regression model fits the observed data.

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

- The R-squared value typically ranges from 0 to 1, although it can be negative for models that fit the data poorly. The range can be interpreted as follows:
  - $R^2 = 0$: The model explains none of the variance in the dependent variable.
    It does not fit the data at all.
  - $R^2 = 1$: The model explains all of the variance in the dependent variable.
    It perfectly fits the data.

How Well Does the Model Fit the data?

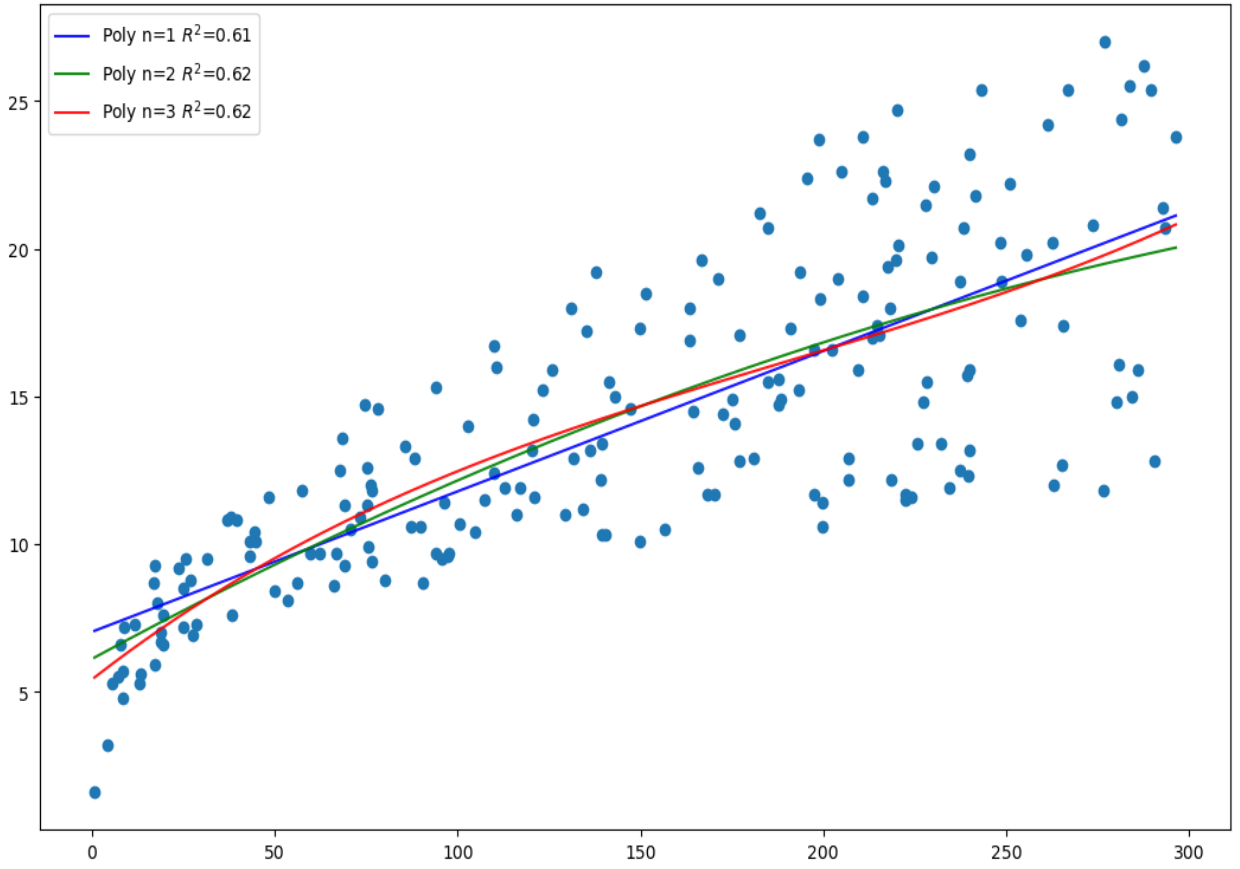
- The R-squared value is a measure of how well a linear regression model explains the variance in the dependent variable. It helps you assess the goodness of fit, with higher values indicating a better fit to the data.



How Well Does the Model Fit the data?

Is that a "good" R-squared value?

- It's hard to say.
- The threshold for a good R-squared value depends widely on the domain.
- Therefore, it's most useful as a tool for **comparing different models**.



Multiple Linear Regression

- Definition: Multiple linear regression uses multiple independent variables to predict the dependent variable.
- Example: Predicting a student's final exam score based on study time, attendance, and previous exam scores.
- Formula: $Y = b0 + b1X1 + b2X2 + ... + bnXn$
- Each x represents a different feature, and each feature has its own coefficient.

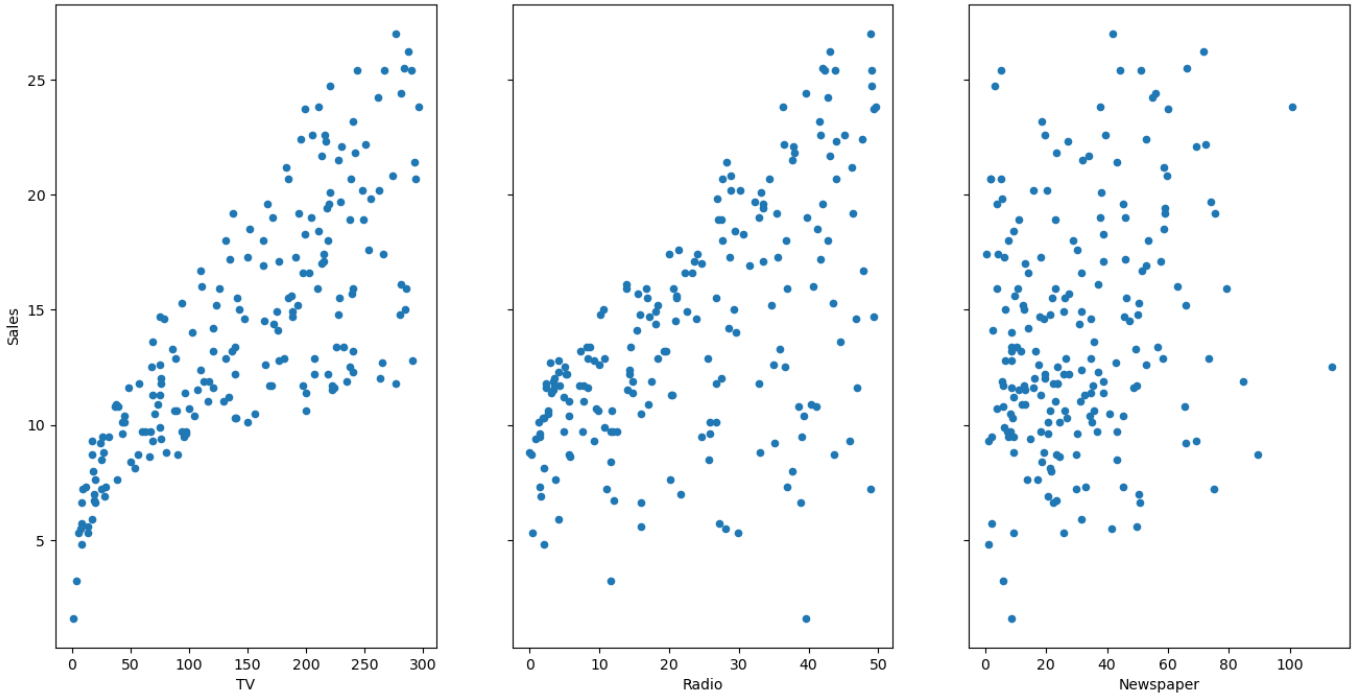
Multiple Linear Regression

- Example: Is there a relationship between ads and sales?
- Formula: Y = b0 + b1 * TV + b2 * Radio + b3 * Newspaper
- Let's use Statsmodels to estimate these coefficients:

```
# create a fitted model with all three features
lm = smf.ols(formula='Sales ~ TV + Radio + Newspaper', data=data).fit()

# print the coefficients
lm.params
```

```
Intercept    2.938889
TV           0.045765
Radio        0.188530
Newspaper   -0.001037
dtype: float64
```



Multiple Linear Regression in scikit-learn

•Let's redo some of the Statsmodels code above in scikit-learn.

```
# create X and y
feature_cols = ['TV', 'Radio', 'Newspaper']
X = data[feature_cols]
y = data.Sales

# follow the usual sklearn pattern: import, instantiate, fit
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(X, y)

# print intercept and coefficients
print(lm.intercept_)
print(lm.coef_)
```

```
2.9388893694594085
[ 0.04576465  0.18853002 -0.00103749]
```

This means the linear regression equation based on the given data would be:

$$\text{Sales} = 2.93888 + 0.04576 \times \text{TV} + 0.18853 \times \text{Radio} - 0.00104 \times \text{Newspaper}$$

```
# pair the feature names with the coefficients
zip(feature_cols, lm.coef_)
```

```
<zip at 0x7fe9880b1b80>
```

```
# predict for a new observation

new_observation = pd.DataFrame({'TV': [100], 'Radio': [25], 'Newspaper': [25]})
predicted_sales = lm.predict(new_observation)

print(predicted_Sales)
```

```
[12.20266701]
```

```
# calculate the R-squared
lm.score(X, y)
```

```
0.8972106381789521
```

Note that **p-values** and **confidence intervals** are not (easily) accessible through scikit-learn.

Feature Selection

- Feature selection, also known as variable selection or attribute selection, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction.
- How do I decide **which features to include** in a linear model?
- Several methods can do feature selection
- If you are unsure which feature selection method to use, you can use cross-validation to evaluate the performance of models trained with features selected by each method.

Feature Selection

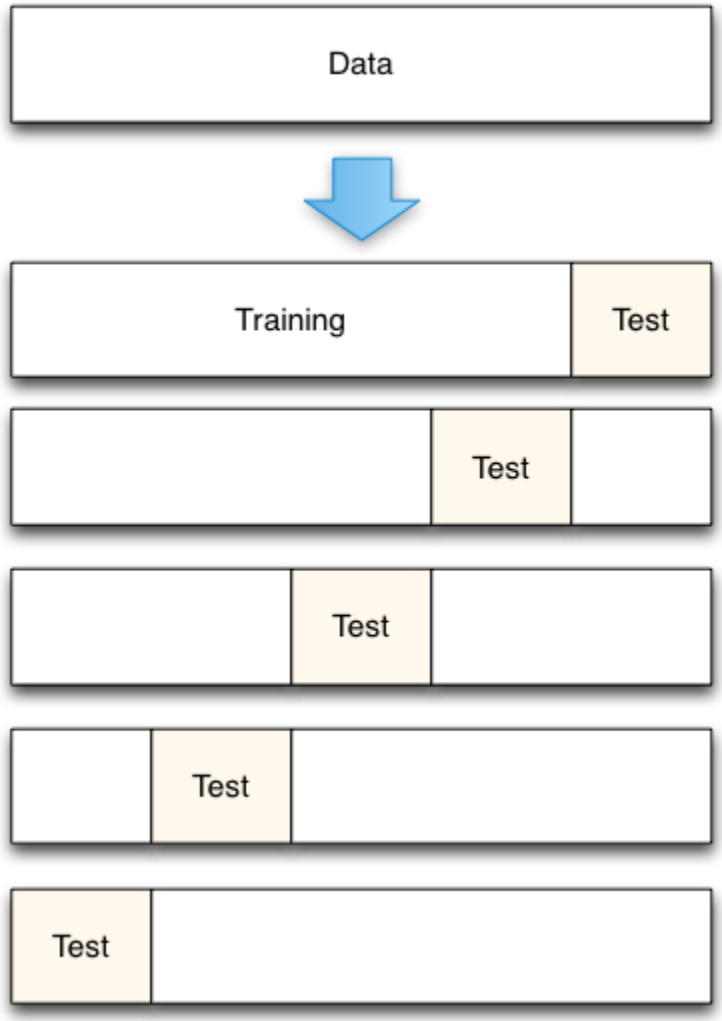
- Cross-validation is a statistical technique used to assess the performance of machine learning models.
- It's particularly useful for gauging how well a model will generalize to unseen data.
- Instead of relying on a single train/test split, cross-validation provides a more comprehensive evaluation by using multiple splits, thereby offering a better estimate of a model's true performance.
- Many techniques including:

  K-Fold Cross-Validation; Stratified K-Fold Cross-Validation; Leave-One-Out Cross-Validation (LOOCV); Time Series Cross-Validation…

Feature Selection

- **5-Fold Cross-Validation (K=5)**

•The most common form of cross-validation.
•The dataset is divided into ‘5' folds.
•The model is trained on ‘4' of those folds and tested on the remaining fold.
•This process is repeated ‘5' times, with each fold serving as the test set exactly once.
•The ‘5' results (e.g., accuracy or mean squared error) are then averaged to produce a single measurement.



Implement CV using scikit-learn

- **Import necessary libraries**, make sure to have the library installed (‘pip install scikit-learn’)
- Load the dataset
- Initialize the model
- Set up 5-fold cross-validation
- Calculate the cross-validation scores
- Print the results.
- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html

Implement CV using scikit-learn

```
from sklearn.model_selection import KFold, cross_val_score
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
```

```
data = load_iris()
X = data.data
y = data.target
```

```
model = LogisticRegression(max_iter = 1000)
```

```
kf = KFold(n_splits = 5, shuffle = True, random_state =42)
```

```
scores = cross_val_score(model, X, y, cv = kf, scoring = 'accuracy')
```

```
print('Accuracy scores:', scores)
print('Mean Accuracy:', scores.mean())
print('Standard deviation of Accuracy:', scores.std())
```

```
Accuracy scores: [1.         1.         0.93333333 0.96666667 0.96666667]
Mean Accuracy: 0.9733333333333334
Standard deviation of Accuracy: 0.024944382578492935
```

Exercise: Predicting House Prices

- **Objective**: Build a linear regression model to predict house prices based on various features and evaluate its performance using 5-fold cross-validation.

1. Dataset Preparation: use the '**California housing'** dataset available in scikit-learn.
2. Exploratory Data Analysis (EDA), check for null values, and visualize some features.
3. Train a Linear Regression Model
4. Evaluate the model using 5-fold cross-validation
5. Further explore: can you try other regression models to improve the performance?

Dataset Preparation:

```
# Import necessary libraries
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split

# Load the dataset
data = fetch_california_housing()
X = data.data
y = data.target

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```