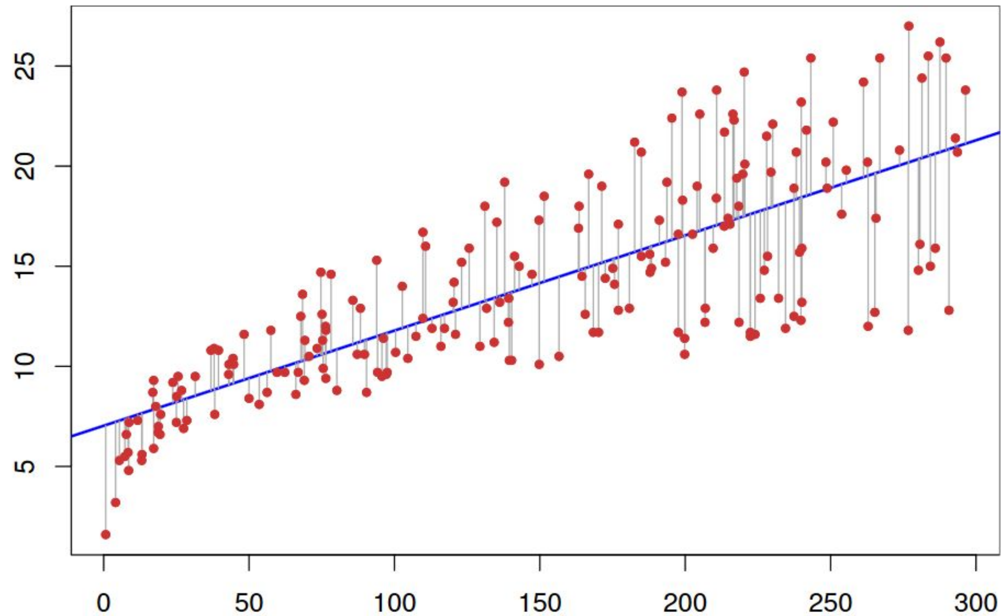
Linear Regression

- Simplă (cea mai)
    - Baseline
    - Underfit ⇔ sub random chance accuracy
- Potrivire dreaptă



- Căutăm o dreaptă

$$Y = b \cdot X + a$$

Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	Price
2010	72000	CNG	Manual	First	26.6 km/kg	998 CC	58.16 bhp	5	1.75
2012	87000	Diesel	Manual	First	20.77 kmpl	1248 CC	88.76 bhp	7	6
2013	40670	Diesel	Automatic	Second	15.2 kmpl	1968 CC	140.8 bhp	5	17.74
2012	75000	LPG	Manual	First	21.1 km/kg	814 CC	55.2 bhp	5	2.35

1. anul fabricației
2. numărul de kilometrii
3. mileage
4. motor
5. putere
6. numărul de locuri
7. numărul de proprietarii (valori între 1 și 4)

8-12. tipul de combustibil - fiind 5 tipuri de combustibil, acesta a fost recodat într-un one-hot vector de 5 componente.
13-14. tipul de transmisie - fiind 2 tipuri de transmisie, acesta a fost recodat într-un one-hot vector de 2 componente. 10 - „Manual”; 01 - ”Automatic”.

X = Km

- X - features
- Y - preţul maşinii (var dep de features)
- a - intercept term, aka val lui y când x=0
- b - panta dreptei
- a, b - coef
    - Random iniţial
    - Învăţaţi ulterior

Cum găsim coef?

MSE - Mean Squared Error

$$J(w) = \frac{1}{n} \sum_{i=1}^{n} (y(x^i) - y^i_{true})^2$$

MAE

Error
- Dif între target (val adev) şi predicted values
- Ridicare la pătrat/abs pt că se pot anula anumite erori

Gradient Descent
- Optimizare
    - Găsire cei mai buni coef aka cel mai bun model

Paşi GD:
1. Alegem random: a, b
2. MSE
3. Folosim alpha - learning rate ca factor de scalare
4. Actualizare coef
5. Repetăm până când MSE nu prea se mai modifică

Multiple Linear Regression

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

More info:
https://towardsdatascience.com/linear-regression-understanding-the-theory-7e53ac2831b5

https://medium.com/datadriveninvestor/understanding-linear-regression-in-machine-learning-643f577eba84

http://onlinestatbook.com/2/regression/intro.html

https://machinelearningmastery.com/linear-regression-for-machine-learning/

https://stattrek.com/regression/slope-test.aspx

Regularizare

Lasso Regression (L1 Regularization) - minimizare eroare şi suma abs a coef.

Ridge Regression (L2 Regularization) - minimizare eroare şi suma pătrată a coef.

```
## Cod dat de load
# load training data
training_data = np.load(os.path.join(path, 'training_data.npy'))
prices = np.load(os.path.join(path, 'prices.npy'))
 # print the first 4 samples
print('The first 4 samples are:\n ', training_data[:4])
print('The first 4 prices are:\n ', prices[:4])
# shuffle
training_data, prices = shuffle(training_data, prices, random_state=0)

# Problema 1
Def normalize(train, test=None):
  # 1. Instantiate the scaler: StandardScaler

  # 2. Fit the training data

  # 3. Transform the training data
```

```
  # 4. If testing is None => return scaled train data
  # 5. Transform also the test data

  # 6. Return scaled train, scaled test


# Problema 2
Def norm_and_train(model, train_samps, train_lbls, test_samps, test_lbls):
  # Normalize
  Train_samps, test_samps = normalize(train_samps, test_samps)
  # Train
  model.fit(train_samps, train_lbls)
  # Predict
  Preds = model.predict(test_samps)
  # MAE, MSE compute
  Mae = mean_absolute_error(test_lbls, preds)
  Mse = # use mean_squared_error

  Return mae, mse

# Split the train set in 3 folds
Train_len = len(train_data)
Samps_per_fold = train_len // 3

# Split in 3 folds
Train_data_1, prices_1 = train_data[:samps_per_fold], prices[: samps_per_fold]
Train_data_2, prices_2 = train_data[samps_per_fold:2*samps_per_fold], …
Train_data_3, prices_3 # 2*samps.. : 3*samps..

# Instantiate the model
Model = LinearRegression()

# Train on each fold (as test)
Mae_1, mse_1 = norm_and_train(model, np.concatenate((train_1, train_2)),
np.concatenate((prices_1, prices_2)), train_data_3, prices_3)

Mae_2, mse_2 = #
Mae_3, mse_3 = #

# print mean mae, mean mse
Mean_mae = (mae_1 + mae_2 + mae_3) / 3

# Problem 3
Alphas = [1, 10,...]
For alpha in alphas:
```

```
  Model = Ridge(alpha)
  # mae_1, mse_1 =
  # mae_2, mse_2 = …
  # …
  # print alpha
  # print mean mae, mean mse

# Problem 4
model = Ridge(alpha=x)
Train_samps = normalize(train_samps)
model.fit(train_samps, prices)
print(‘coefs: ‘, model.coef_)
print(‘bias: ‘, model.intercept_)


Most_sign_feat = np.argmax(np.abs(model.coef_)) + 1

Least_sign_feat =
```