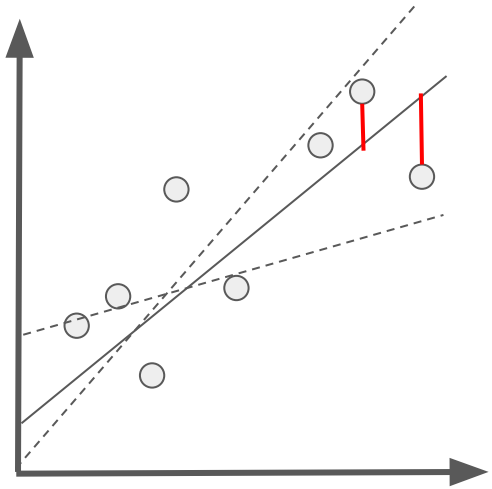
5장 회귀

신림프로그래머 최범균

회귀(regression)

- 여러 개의 독립변수와 한 개의 종속변수 간의 상관관계를 모델링
- $Y = W_1 * X_1 + W_2 * X_2 + \cdots W_n * X_n$
  - Y : 종속 변수(결정 값)
  - X_i : 독립 변수(피처)
  - W_i : 회귀 계수(Regression coefficients)
- 회귀의 핵심은 피처와 결정 값 데이터를 학습해서 최적의 회귀 계수를 찾는 것
- 책에서 소개하는 회귀 모델
  - 일반 선형 회귀
  - 규제 적용 모델: 릿지(Ridge), 라쏘(Lasso), 엘라스틱넷(ElasticNet)
  - 로지스틱 회귀(Logistic Regression)

선형 회귀 이해



회귀 모델: $y = w_0 + w_1 * x$

오류
- 실제 값과 회귀 모델의 차이
- 남은 오류의 의미로 잔차라고도 함

최적의 회귀 모델 찾기
- 오류 합을 최소화하는 회귀 계수를 찾는 것

오류 합
- RSS(Residual Sum of Square) : 오류 값 제곱을 더하는 방식
- MAE(Mean Absolute Error) : 오류의 절댓값을 더하는 방식

회귀에서 오류 합을 구하는 함수를 비용(cost) 함수 또는 손실(loss) 함수라고 함

비용 최소화하기
- 경사 하강법(Gradient Descent) 이용
- 성능 위해 확률적(Stochastic) 경사 하강법

참고 자료:
- 김성훈 교수님 자료 : https://youtu.be/TxIVr-nk1so

사이킷런 선형 회귀 코드 예

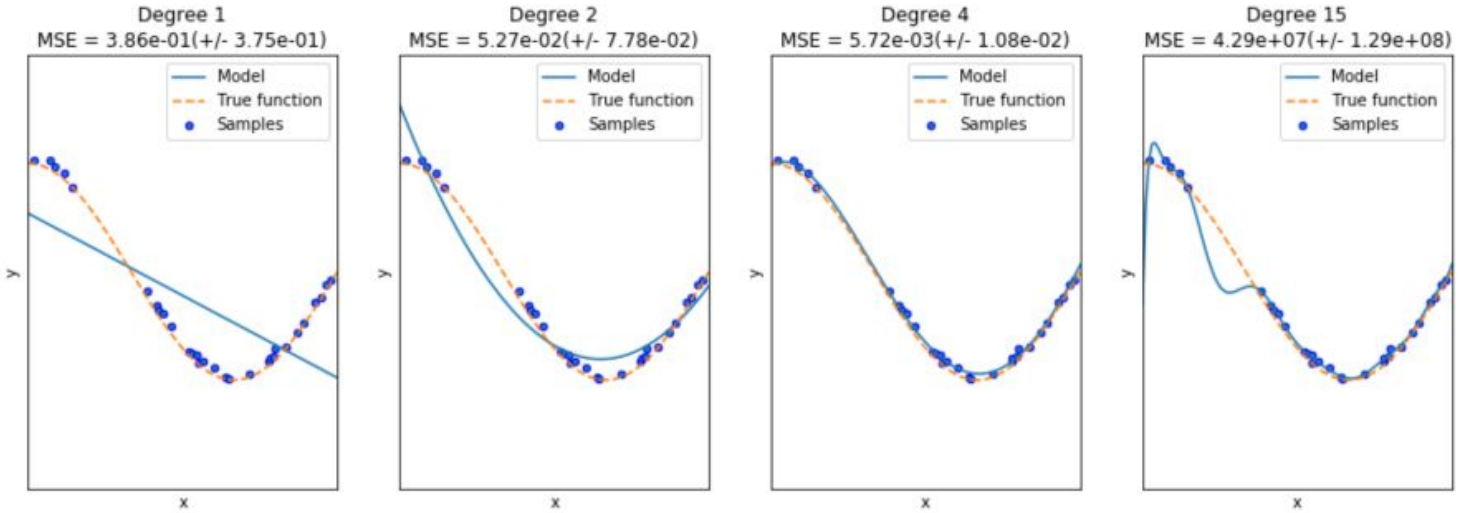
```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(X_train, y_train)
y_preds = lr.predict(X_test)

mse = mean_squared_error(y_test, y_preds)
rmse = np.sqrt(mse)

print('MSE : {0:.3f}, RMSE: {1:.3f}'.format(mse, rmse))
print('Variance score: {0:.3f}'.format(
      r2_score(y_test, y_preds)))
print('절편 : ', lr.intercept_)
print('회귀 계수 : ', np.round(lr.coef_, 3))
```

[사이킷런 지원 평가 지표]

평가 지표	설명	사이킷런 함수	스코어 함수
MAE (Mean Absolute Error)	오류 절댓값의 평균	mean_absolute_error	'neg_mean_absolute_error'
MSE (Mean Squared Error)	오류 제곱의 평균	mean_squared_error	'neg_mean_squared_error'
R^2	실제 값 분산 대비 예측값 분산 비율	r2_score	'r2'

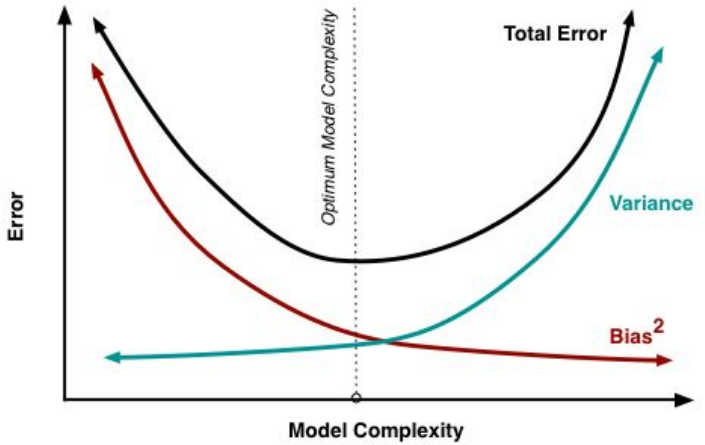
다항 회귀

- 다항 회귀: 회귀가 독립 변수에 대한 다항식으로 표현
  - 예 : $y = w_0 + w_1*x_1 + w_2*x_2 + w_3*x_1*x_2 + w_4*x_1^2 \cdots$
- 사이킷런은 PolynomialFeatures 클래스로 피처를 다향식 피처로 변환
- 차수에 따른 과소적합/과적합 가능성



편향-분산(Bias-Variance) 트레이드 오프

- 고편향 : 한 방향성으로 치우침
- 고분산 : 지나치게 높은 변동성
- 높은 편향/낮은 분산 → 과소적합되기 쉬움
- 낮은 편향/높은 분산 → 과적합되기 쉬운



출처: http://scott.fortmann-roe.com/docs/BiasVariance.html

규제 선형 모델

- 데이터에 적합하면서도 회귀 계수가 급격히 커지는 것을 제어해야 함
- 회귀 계수의 크기를 제어하기 위해 비용 함수에 규제(regularization) 추가
- 릿지: L2 규제
  - 비용 함수 = RSS(W) + alpha * $||W||_2^2$
  - alpha 값이 커지면 회귀 계수 W의 값을 작게 해 과적합 개선
  - alpha 값이 커지면 회귀 계수 W가 커져도 비용이 증가해 어느 정도 상쇄 가능
- 라쏘: L1 규제
  - 비용 함수 = RSS(W) + alpha * $||W||_1$
  - 불필요한 회귀 계수를 급격히 감소시켜 0으로 만들고 제거 (피처 선택의 특성)
- 엘라스틱넷 회귀 : L2 규제와 L1 규제를 결합

선형 회귀 모델과 데이터 변환

- 피처값과 타깃값의 분포가 정규 분표 형태일 때 성능 좋음
  - 특정값의 분포가 치우친 왜곡 형태인 경우 성능에 부정적 영향
- 왜곡이 심할 경우 피처/타깃값에 대해 변환 작업 수행
- 주요 변환 방법
  - StandardScaler, MinMaxScaler 등 표준화, 정규화
    - 성능 향상 없으면 추가로 다항 특성 적용
  - 로그 변환 : 선형 회귀에서는 로그 변환을 많이 사용
    - 일반적으로 타깃값은 원본으로의 원복을 고려해 로그 변환 적용

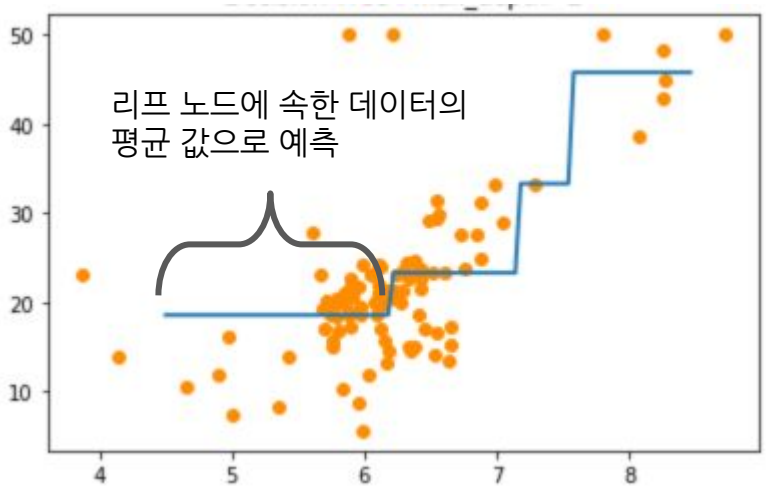
로지스틱 회귀

- 시그모이드(Sigmoid) 함수 최적선을 찾아 이 함수의 값을 분류의 확률로 사용
  - 시그모이드 함수 : $y = 1 / (1 + e^{-x})$
- 가볍고 빠르면서도 이진 분류 예측 성능이 뛰어남

```
# 사이킷런 예제 코드
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV

lr_clf = LogisticRegression()

# penalty : 규제 유형, C : 규제 강도 (C = 1/alpha) C 값이 작을수록 규제 강도 큼
params={'penalty': ['l2', 'l1'], 'C': [0.01, 0.1, 1, 5, 10]}
grid_clf = GridSearchCV(lr_clf, param_grid=params, scoring='accuracy', cv=3)
grid_clf.fit(data_scaled, cancer.target)
print('최적 하이퍼 파라미터:{0}, 최적 평균 정확도:{1:.3f}'.format(
    grid_clf.best_params_, grid_clf.best_score_))
```

회귀 트리

- 리프 노드에 속한 데이터 값의 평균값을 구해 회귀 예측값을 계산
- 사이킷런 : 결정트리, 랜덤 포레스트, GBM 용 회귀 트리 Estimator 제공



데이터 처리 예: 캐글 자전거 대여 수요 예측 데이터

- 데이터 형태
  - CSV
  - 첫 칼럼(datetime)은 문자열 값

```
datetime,season,holiday,workingday,weather,temp,atemp,humidity,windspeed,casual,registered,count
2011-01-01 00:00:00,1,0,0,1,9.84,14.395,81,0,3,13,16
```

- 첫 칼럼 DataFrame 로딩시 object로 인식 → datetime 타입으로 변환

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
datetime      10886 non-null object
```

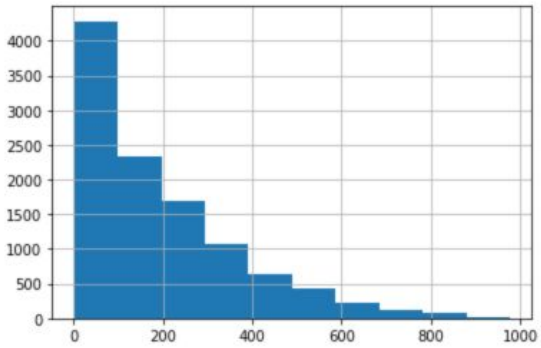
```
bike_df = pd.read_csv("./bike_train.csv")
bike_df.info()

bike_df['datetime'] = bike_df.datetime.apply(pd.to_datetime)
```

```
Data columns (total 16 columns):
datetime      10886 non-null datetime64[ns]
```

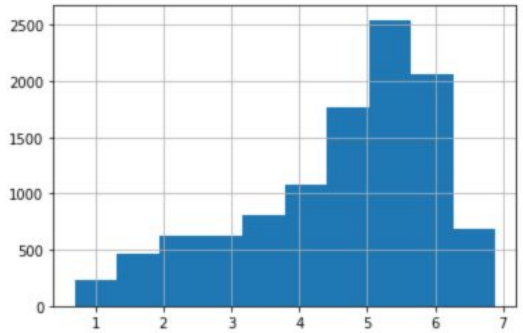
```
bike_df.info()
```

데이터 처리 예: 캐글 자전거 대여 수요 예측 데이터

- 카테고리 값 인코딩 처리
  - 회귀에서 년, 월, 일, 휴일 여부 등 카테고리 값은 원-핫 인코딩 처리
  - 코드 예:
    - ohe = pd.get_dummies(X_features, columns=['year', 'month', 'hour', …생략])
- 왜곡된 타킷 값 로그 변환처리




```
y_log_trans = np.log1p(y_target)
```



데이터 처리 예: 캐글 주택 가격

- 피처 타입, Null 데이터 확인

```
print('전체 피처 type:\n', house_df.dtypes.value_counts())
isnull_series = house_df.isnull().sum()
print('\nNull 칼럼과 건수:\n',
    isnull_series[isnull_series > 0].sort_values(ascending=False))
```

```
데이터 세트 shape: (1460, 81)
전체 피처 type:
 object     43
int64       35
float64      3
dtype: int64

Null 칼럼과 건수:
 PoolQC          1453
MiscFeature      1406
Alley            1369
Fence            1179
FireplaceQu       690
LotFrontage       259
GarageYrBlt        81
GarageType         81
GarageFinish       81
GarageQual         81
GarageCond         81
BsmtFinType2       38
BsmtExposure       38
BsmtFinType1       37
BsmtCond           37
BsmtQual           37
```

- Null이 많은 피처는 드롭(숫자형 피처는 평균값 대체)

```
house_df.drop(['PoolQC', 'MiscFeature', 'Alley', 'Fence',
'FireplaceQu'], axis=1, inplace=True)
house_df.fillna(house_df.mean(), inplace=True)
```

- 문자형 피처 : 원-핫 인코딩

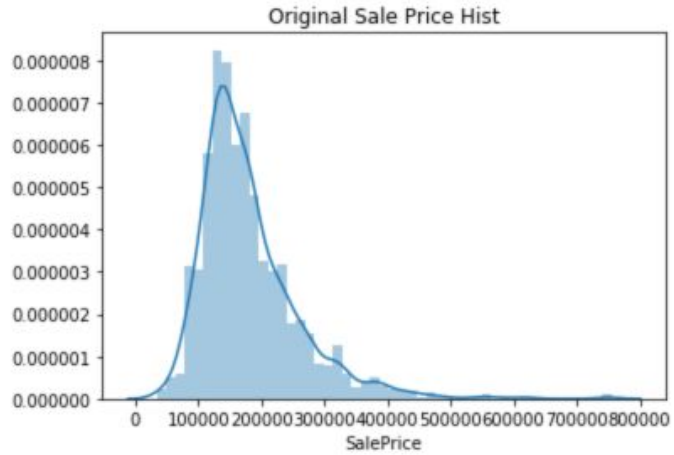
```
house_df_ohe = pd.get_dummies(house_df)
```

데이터 처리 예: 캐글 주택 가격

- 타깃 값 분포 왜곡 처리 : 로그

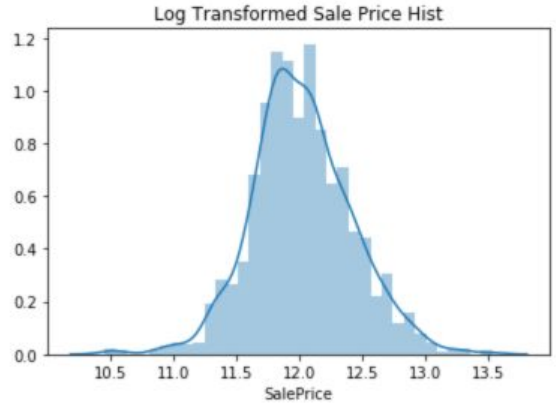
```
plt.title('Original Sale Price Hist')
sns.distplot(house_df['SalePrice'])
```

```
<matplotlib.axes._subplots.AxesSubplot a
```




```
log_SalePrice = np.log1p(house_df['SalePrice'])
plt.title('Log Transformed Sale Price Hist')
sns.distplot(log_SalePrice)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x13c6
```



데이터 처리 예: 캐글 주택 가격

- 피처 중에서 왜곡이 큰 피처를 정리(로그 사용)

```
from scipy.stats import skew

features_index = house_df.dtypes[house_df.dtypes != 'object'].index
skew_features = house_df[features_index].apply(lambda x : skew(x)) #
skew_features_top = skew_features[skew_features > 1]
print(skew_features_top.sort_values(ascending=False))

house_df[skew_features_top.index] = np.log1p(house_df[skew_features_top.index])
```

- 정리 후 재학습

데이터 처리 예: 캐글 주택 가격

- 회귀 계수가 높은 피처는 이상치 데이터 확인 후 제거
  - 예, GrLiveArea 피처