Apuntes Modulo 4 a00178679

Para realizar un mejor modelo se usan los feactures, para obtener predicciones

Diferentes modelos

Simple – SLR, relación entre 2 variables, predicciones x y target y

Se define la x y y se usa el LM.fit

Para obtener el target se usa el lm.intercept\_+ lm.coef\_\* x

Multiple – un target vs varias variables, se usan las mimsas formulas o coeficientes que ne la simple

Polinomial

Visualizacion

Regresión ploy

Regplot from seaborn

Sns.regplot(x=feature, y =target, data=df)

Plt.ylim(0,)

Ax1 = sns.distplot(df[targey],hist=false, color=”r”, label=”actual”)

Sns.distplot(Yhat, hist=false, color=”b”, label=”fitted”, ax=ax1)

Polinomial

Pueden ser cuadratica, cubica o de orden superior

F=np. polyfit(x,y,”power”)

Para crear el fit no se puede usar el de numpy

Se usa sklearn.preprocessing import PolynomialFeatures

Pr=PolynomialFeatures(degree=”power”, include\_bias=false)

X\_polly=pr.fittransform(x[[[‘col1’,’col2’,’’]])

Tambien al usar varias variables podemos usar sklearn.preprocessing import StandarScaler

SCALE = StandarScaler()

Scale.fit(x\_data[[‘’,’’,’’,]])

X\_scale = SCALE.transform(x\_data[[‘’,’’,’’,]])

Tambien podemos usar pipelines

Sklearn.pipeline import Pipeline

Input = [(‘scale’, StandarScaler()), (‘polynimial’,PolynomialFeatures(degree=2)),(‘model’,LinearRegression())]

pipe=Pipeline(Input)

Pipe.Train(x[‘’,’’,’’,’’],y)

Yhat=Pipe.predict(x,y)

Con lo pipeliens simplifocamos el proceso haciendo en un solo paso las transformación y normalizaciones

Como evaluar numéricamente los modelos

MSE, Error medio cuadrado, entre un valor cercano a cero es mejor

Rcuadrada = lm.Score si es negativo hay overfitting

Predicciones y toma de decisiones

Np.arange(inicio,fin,steps)

No siempre un MSE menor implica mejor modelo