# Linear Regression

## Learning Portfolio 2

universität innsbruck

# Data Set

---

The data set is a data set retrieved on kaggle. It displays weekly sales data for numerous Walmarts from 2010 - 2012.

The goal of the data set is to predict weekly sales data.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Store         6435 non-null   int64  
 1   Date          6435 non-null   object 
 2   Weekly_Sales  6435 non-null   float64
 3   Holiday_Flag  6435 non-null   int64  
 4   Temperature   6435 non-null   float64
 5   Fuel_Price    6435 non-null   float64
 6   CPI           6435 non-null   float64
 7   Unemployment  6435 non-null   float64
dtypes: float64(5), int64(2), object(1)
memory usage: 402.3+ KB
```

universität innsbruck

# Data Preprocessing

- Extract year from date
- Extract calendar week from date
- Calculate previous weeks' sales volume
- Scaling and transformation

```
default_num_pipeline = make_pipeline(StandardScaler())

log_pipeline = make_pipeline(
 FunctionTransformer(np.log),
 StandardScaler())

label_pipeline = make_pipeline(
 FunctionTransformer(np.log))

cat_pipeline = make_pipeline(OneHotEncoder(handle_unknown="ignore"))

preprocessing = ColumnTransformer([
 # scaling for non skewed attributes
 ("normal", default_num_pipeline, ["Temperature"]),
 # log and scaling for skewed/multi-model attributes
 ("log", log_pipeline, ["Fuel_Price", "CPI", "Unemployment"]),
 # one-hot-encoding for week, year and store
 ("cat", cat_pipeline, ["Week", "Year", "Store"]),
 # Log for Prev_Week_Sales
 ("sales", label_pipeline, ["Prev_Week_Sales"]),
 ])

preprocessing_label = ColumnTransformer([
 # Log for Weekly_Sales
 ("log", label_pipeline, ["Weekly_Sales"])
 ])
```

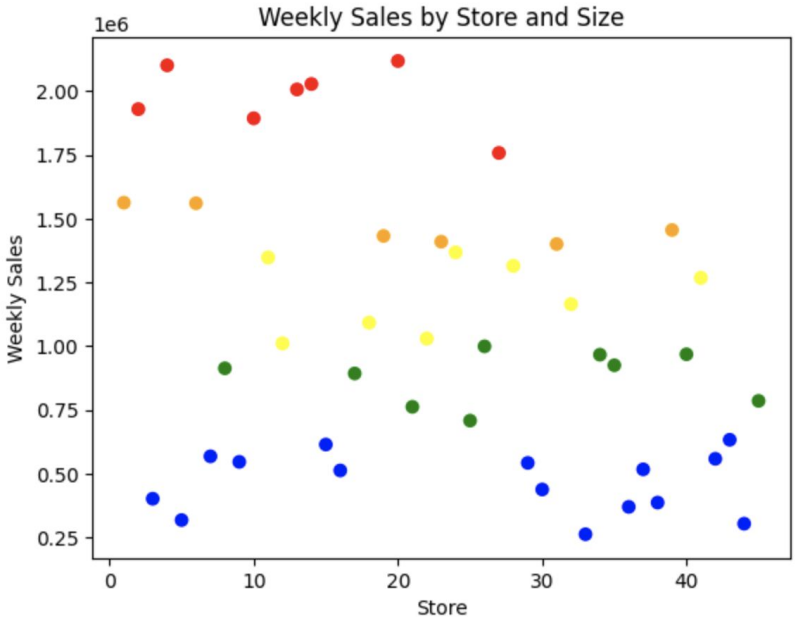
```
[37] # Transform dates to calendar week
     data['Week'] = pd.to_datetime(data["Date"]).dt.isocalendar().week
     data['Year'] = pd.to_datetime(data["Date"]).dt.isocalendar().year
     data = data.drop("Date", axis=1)

     # We got Time Series Data. So we are adding the previous week sale for each store to the row
     # As will be shown later, it is a really good predictor for this week's sale.
     def calculate_previous_week_sales(df):
       df = df.sort_values(['Year', 'Week'], ascending=True)
       df['Prev_Week_Sales'] = df.groupby('Store')['Weekly_Sales'].shift(1)
       mean_sales = df.groupby('Store')['Weekly_Sales'].mean()
       df['Prev_Week_Sales'] = df.apply(lambda row: mean_sales[row['Store']] if pd.isna(row['Prev_Week_Sales']) else row['Prev_Week_Sales'], axis=1)
       return df

     data = calculate_previous_week_sales(data)
```

universität innsbruck

# Data Visualization

---

- In the data, some stores sell below 500k on average, while others sell above 2M on average.
- Hypothesis: Different stores behave differently, thus store IDs were one-hot encoded in this case.

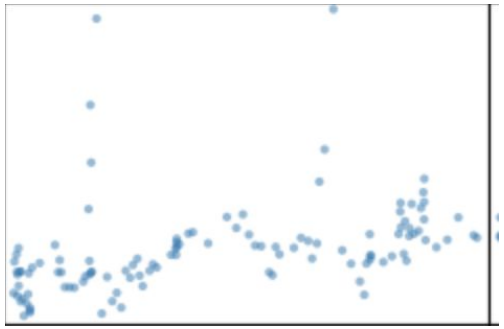



universität innsbruck

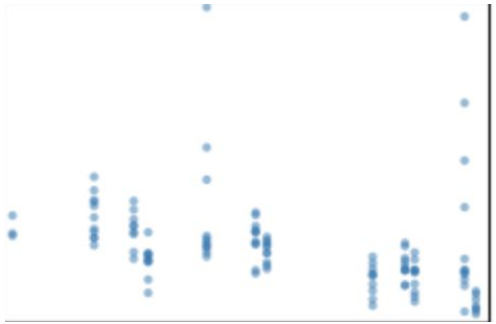
# Data Visualization

---

- CPI has a positive, non-linear relationship with sales
- Unemployment has a negative relationship with sales



CPI (X) vs. Sales (Y)



Unemployment (X) vs. Sales (Y)

universität innsbruck

# Training

---

- 10-fold cross-validation
- Measure: RMSE
- LinearRegression: ca. 10 % off median sales
- Random Forest: ca. 85 % off median sales

```
[48] pd.Series(lin_rmses).describe()

count        10.000000
mean      94162.139880
std        6043.175273
min       83478.295877
25%       91265.900874
50%       93715.000363
75%       97603.019447
max      103776.133297
dtype: float64
```

```
pd.Series(forest_rmses).describe()

count        10.000000
mean     789770.214784
std       14639.449646
min      764505.493718
25%      785229.516481
50%      790403.988053
75%      797108.642694
max      810472.039417
dtype: float64
```

universität innsbruck

# Testing

- RMSE: ca. 10 % off on average
- MAE: < 5 % off on average
- Median of predictions is very close to the median of actual sales values

```
[50] from sklearn.metrics import mean_squared_error
     from sklearn.metrics import median_absolute_error

     X_test = test_set.drop(["Weekly_Sales"], axis=1)
     y_test = test_set[["Weekly_Sales"]].copy()

     test_label_transformed = preprocessing_label.fit_transform(y_test)
     test_prepared = preprocessing.fit_transform(X_test)
     final_predictions = lin_reg.predict(test_prepared)

     rmse = mean_squared_error(np.exp(test_label_transformed), np.exp(final_predictions), squared=False)
     mae = median_absolute_error(np.exp(test_label_transformed), np.exp(final_predictions))

     median_pred = np.median(np.exp(test_label_transformed))
     median_actual = np.median(np.exp(final_predictions))

     print(rmse)
     print(mae)
     print(median_pred)
     print(median_actual)

     91330.89270560814
     38613.07987333543
     966817.2400000002
     962626.9938374124
```

universität innsbruck

# Kontakt

**Fabian Leuk**
12215478
fabian.leuk@student.uibk.ac.at