Jessica Kirkham
Data Fundamentals
DT401

Data Fundamentals: the breakdown of the roles and functions of technology solutions and the ethics and systems behind them.

Layout:

Explain the roles and functions of technology solutions within an organisation.

Technology solutions, fundamental to any organisation, allows for fast and efficient decision-making. A function of technology solutions, Business Intelligence, enables data-driven decision-making, utilising sources like financial systems and customer relationship management systems within a unified database. An example is Inventory Management Systems, a set of procedures placed to reduce cost and waste (Mbanje & Lunga, 2016), by managing customer orders, stock levels, and predict future stock levels by forecasting trends. The high-street retailer H&M uses RFID tags to do this (Mixson, 2021).

Using Business Intelligence improves operational efficiency, as operations are automated and streamlined. Morrisons' partnership with Blue Yonder is an example of this, providing a service using artificial intelligence to predict stock level trends. This service completes 430 million calculations and 13 million decisions per-day regarding which stock to purchase (Roberts, 2017), leading to food waste and cost reduction. Using a technological solution to improve operational efficiency will inherently reduce waste in resources, cost or time.

Customer satisfaction makes any organisation successful and by using a technology solution, such as focusing on customer experience, an organisation can continue to profit. This technique can be seen through the expansion of fashion brand 'Zara' and the use of self-checkouts. This transition saves time for the customers. However, some argue there are limitations to the use of this as it could lead to fast fashion and loss of jobs which raises ethical questions.

Three role categories can be used to create and implement technology solutions: Strategic, Developmental and Analytical. Strategic roles require leadership and decision-making skills to lead innovation. Dr Werner Vogels, the Chief Technology Officer of Amazon, developed AWS, Amazon Web Services (AWS, 2021), allowing Amazon to explore new markets and increase profits. Another strategic role is an IT project manager, which requires strong leadership, communication, and time management skills. It is about planning and execution; ensuring a project meets deadlines and stakeholders' expectations.

Development roles design and maintain technology solutions, which require strong technical skills. An example is a DevOps Engineer. Within this role, collaboration skills are essential as they are responsible for the innovation of IT services and solutions by embracing agile methodologies (Flemming, 2017).

Finally, Analytical roles focus on using data to drive operations by providing suitable solutions, such as a Data Analyst. High-level data skills are needed to collect, process, and manipulate data to gain insights to drive business intelligence and decision-making. Within my organisation, this role shows inefficient areas of IT services.

In conclusion, technical solutions can be spread across functions and roles that must work together to increase reliability, efficiency and customer experience.

Critically reflect upon data management systems and their role in digital and technology solutions.

Data management methods must be evaluated in terms of data security and efficiency, as, if these factors are not prioritised, these systems will be ineffective.

When considering security, the “six dimensions of data quality” should be prioritised. These ask organisations if their data is: valid, unique, complete, timely, accurate and consistent. Although these focus on how fit for purpose data is by completing data quality audits, organisations can ensure the data they are using is free from bias and is complete and accurate for the purpose collected (Hub, 2021). Using the “six dimensions”, an organisation can ensure it is complying with GDPR regulations and ensuring data is accurate and consistent. However, it is crucial data quality audits are completed utilising automated tools, as the complexity can be time consuming and lead to human error (Hub, 2021).

Another framework that prioritises security and efficiency is the “Data Lifecycle”, how organisations should approach data through its lifetime. Managing data creation, storage, archival and deletion, provides clarity on practices which can prevent GDPR violations and fines (IBM, no date). However, this management system can be ineffective within solutions as it can be challenging meeting criteria for all lifecycle stages and ensuring all personnel understand all policies effectively. This could lead data to be mishandled.

Utilising data principles is essential for efficiency, when regularly repurposing the collection and integration of data to support data-driven decisions. Most importantly, real-time processing of data using: ACID principles (ensuring numerous users can access a database simultaneously, whilst remaining consistent), ETL (Extract Transform Load) and ELT (Extract Load Transform) methods. These allow transactions to be isolated allowing the database to be continuously functional. ETL methods help streamline integration by creating a fixed schema. However, new information can produce a bottleneck. ELT methods allow multiple users to access raw data without directly accessing each source, improving data security. However, delaying data transformation can lead to complex and expensive transformations.

“FAIR principles” improves technology solutions by enhancing data sharing, allowing data to be used responsibly as it becomes more; findable, accessible, interoperable and reusable. Data becomes useful for different purposes leading to increased productivity, decision-making and better outcomes.

Finally, effective use of data structures, such as tables, arrays and key-value pairs, enhances data management systems. Tables allow easy, organised data manipulation giving efficient searching and sorting because of the specific fields. However, they are memory intensive as the data amount increases and it becomes hard to represent complex relationships. An array is an alternative, using less memory and is easily searchable. But they are immutable, and unsuitable for new data; all array data must be related. Trees can show hierarchal, non-linear relationships between data, which are flexible and quickly searchable. However, they must be balanced and can be memory intensive as data increases. A final alternative is Key-Value Pairs, which provide security and quick searching as it creates a “dictionary” for searching key-related data. However, they have a complex set up.

In conclusion, for technology solutions to be effective, data management systems must be specifically thought out for an organisation with a clear strategy protecting sensitive data and providing consistent functionality.

Select and apply appropriate data gathering methods for a given data analysis scenario.

I have created a regression model using the Seaborn dataset.

GATHERING DATA
The first stage: gather data from the pre-defined dataset 'taxis', collected through surveys within March and April 2019 (Abdullah, 2023). Acknowledge how the sample has been collected to understand whether it is truly representative of the population.

To do this, the dataset and relevant python packages must be imported into my project. These packages are pandas, scikit-learn and statsmodels.api.These are used to: manipulate, visualise and model data.

Refer to GUIDED DATA PROJECT kernel 12

DATA PREPARATION
This next stage is understanding the different variables, how they interact and needed to ensure we get the expected data. For this project, it has been done before being imported but it's important to look at the type of data within the table.

Refer to kernel 17

DATA EXPLORATION
Then, data is explored to identify relationships between variables - to develop a hypothesis for the model to test. Identifying outliers and checking data quality is crucial. The min and max values are studied to see if the correlation is what is expected using prior-known knowledge.

Refer to kernel 27

This can also be done through visualising data using histograms and scatter plots, which clearly shows the strength of correlation.

Refer to kernel 36

However, with many visualisations displayed, it is hard to clearly differentiate the points of correlation. Instead, a pandas method can be used, creating a table to show correlation numerically. 1 is strong correlation; 0 is weak.

Refer to kernel 40

CREATING A HYPOTHESIS

A hypothesis is a testable statement of what we expect an outcome of a particular model to be. Previously, it was outlined *passengers, distance* and *tolls* were strongly correlated. Therefore, these will be used in the hypothesis.

H0: the greater the distance and the more tolls taken will increase the fare
H1: the distance does not impact the fare / the number of tolls does not increase the fare

Dependant: fare
Independent: distance, tolls

BUILDING THE MODEL

The first step is splitting the data for training and testing. The test size is 20% of the data. The random state is 42 to prevent the data from looking different each time the model is run. On a larger scale, the initial test data would create an initial state of the model. The model would go through iterations until it has a better fit to the data. Sometimes the models can be overtrained, where it becomes 'overfitted' (IBM,2023).

Then, the Ordinary Least Squares function is used to construct the model and a summary is shared.

Refer to kernel 55

EVALUATING THE MODEL

Here, test data is used to produce a scatter graph showing the line of best fit compared to the correlation of test data. The R-Value is also shown. It's important to compare data points (R-Values) from the predictions to the test data.

Refer to kernel 66

RESULT

Looking at the scatter graph, the R-Value and comparing it with the significance levels, H0 can be accepted as it shows a direct correlation between fare, distance and tolls.

Regression template - Seaborn datasets

November 18, 2024

1 Regression template for Seaborn example datasets

1.1 Task

We are going to build a linear regression model and evaluate its performance. For a regression we need:

- A dependent variable. This is the one we will predict. For a regression model this needs to be a continuous variable.
- One or more independent variables. These are the ones we are using to predict the dependent variable. For a regression all of these need to be numerical and at least some need to be continuous.

For a linear regression model, our hypothesis is that our dependent variable:

- Is influenced by each of the independent variables: the independent variable affects the de- pendent
- Has a relationship with each independent variable that is linear
- Has a similar distribution to each of the independent variables
- Is influenced more strongly by each independent variable than the influence they have on each other

## Contents In this notebook, we will:

- Import packages and load in some data
- Prepare the data so we can explore it
- Explore the data and make our testable hypothesis
- Split the data for test and train
- Build the model
- Interpret the model results

Reminder: You don't need to understand all the code here for now, just

look for: What we are trying to do with each code cell. How does it fit in

our objective?

What the outputs of each code cell tell us? Are we reading too much into the results of each code cell?

Some of the code cells will have parts that you will need to change to fit your data. You will be told what to change in the comment before the code cell.

## Import packages and read data Back to Contents

Let's start by importing the Python packages we will need: - **pandas**: a tabular data manipulation package - **seaborn**: a data visualisation package - **scikit-learn**: a model building package - **statsmodels.api**: a model building package

[10]:

```
import pandas as pd
import seaborn as sns
import statsmodels.api as sm

from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error

sns.set_style("whitegrid")
```

We will be using one of these Seaborn example datasets:

- taxis: New York taxi journeys
- titanic: Records of details of passengers on the Titanic
- tips: Restaurant bills and tips
- penguins: Physical details of various penguins
- iris: Measurements of different Iris flowers

These can be loaded using the Seaborn function load_dataset. We can have a look at the first few rows using the method head

[12]:

```
data = sns.load_dataset('taxis')
data.head()
```

[12]:

```
               pickup              dropoff  passengers  distance  fare   tip  \
0 2019-03-23 20:21:09  2019-03-23 20:27:24           1      1.60   7.0  2.15
1 2019-03-04 16:11:55  2019-03-04 16:19:00           1      0.79   5.0  0.00
2 2019-03-27 17:53:01  2019-03-27 18:00:25           1      1.37   7.5  2.36
3 2019-03-10 01:23:59  2019-03-10 01:49:51           1      7.70  27.0  6.15
4 2019-03-30 13:27:42  2019-03-30 13:37:14           3      2.16   9.0  1.10

   tolls  total   color      payment            pickup_zone  \
0    0.0  12.95  yellow  credit card        Lenox Hill West
1    0.0   9.30  yellow         cash  Upper West Side South
2    0.0  14.16  yellow  credit card          Alphabet City
3    0.0  36.95  yellow  credit card              Hudson Sq
4    0.0  13.40  yellow  credit card           Midtown East

            dropoff_zone pickup_borough dropoff_borough
0    UN/Turtle Bay South      Manhattan       Manhattan
1  Upper West Side South      Manhattan       Manhattan
2           West Village      Manhattan       Manhattan
3         Yorkville West      Manhattan       Manhattan
4         Yorkville West      Manhattan       Manhattan
```

** MLO 3 Comments: The first 5 rows have been displayed. Using a table allows me to see different variables in the dataset. It also shows the data type of each variable.

### Prepare the data Back to Contents

For this notebook we are assuming the data has been prepared before being loaded in. However, it is always important to check that you have what you expect.

We can look at what kind of data our table contains using the info method.

We should be asking ourselves:

- Does the data contain the columns I expect?
- Do the columns have the data type I would expect?
- Do any of the columns have missing values? Are these in any columns we intend to use? There cannot be any null values in the rows we plan to use in our model.

[17]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6433 entries, 0 to 6432
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   pickup           6433 non-null   datetime64[ns]
 1   dropoff          6433 non-null   datetime64[ns]
 2   passengers       6433 non-null   int64
 3   distance         6433 non-null   float64
 4   fare             6433 non-null   float64
 5   tip              6433 non-null   float64
 6   tolls            6433 non-null   float64
 7   total            6433 non-null   float64
 8   color            6433 non-null   object
 9   payment          6389 non-null   object
 10  pickup_zone      6407 non-null   object
 11  dropoff_zone     6388 non-null   object
 12  pickup_borough   6407 non-null   object
 13  dropoff_borough  6388 non-null   object
dtypes: datetime64[ns](2), float64(5), int64(1), object(6)
memory usage: 703.7+ KB
```

** MLO3 Comments: This table shows there are 14 variables and 6433 bits of data. There are no null values meaning all variables have associated data. However, not all varibales have a complete set of data. payment, pickup_zone, dropoff_zone, pickup_borough and dropoff_borough have less than 6433 values. This is important as if our hypothesis used these variables there wouldn't be an accurate dataset to create and test a model.

If we have any columns with null values that we want to use in our model, we will need to drop those pieces of data. Pandas gives us the dropna method to do this. If you only need to drop null values from one column you can use the `subset` parameter to pass a list of the columns you wish to be checked for null values.

This piece of code has been commented out, so it will not run. To use this, remove the # from the start of the line and replace `'variable name'` with the column that you wish to be checked for null values. You can examine more than one column at a time by listing all the columns in the square brackets.

[21]:
```
# data = data.dropna(subset=['variable name', ])
```

If one of the columns is only useful as a unique row identifier, we can use it as an index. We can set it using the `set_index` method. Replace `'index column name'` with the column that you wish to set as the index. If you need this, remove the # to uncomment the code.

[24]:
```
#data = data.set_index('ID column name')
```

### Exploring the data Back to Contents

Now that we have checked our data is clean, we can explore it. A good starting point is to look at the descriptive statistics and check that they seem reasonable. We can do so using the `describe` method.We should be asking ourselves:

- Does the data have unexpected outliers (looking at the max and min values) that might suggest a data quality issue?
- Is the spread of the data what you would expect

```
data.describe()
```

[27]:

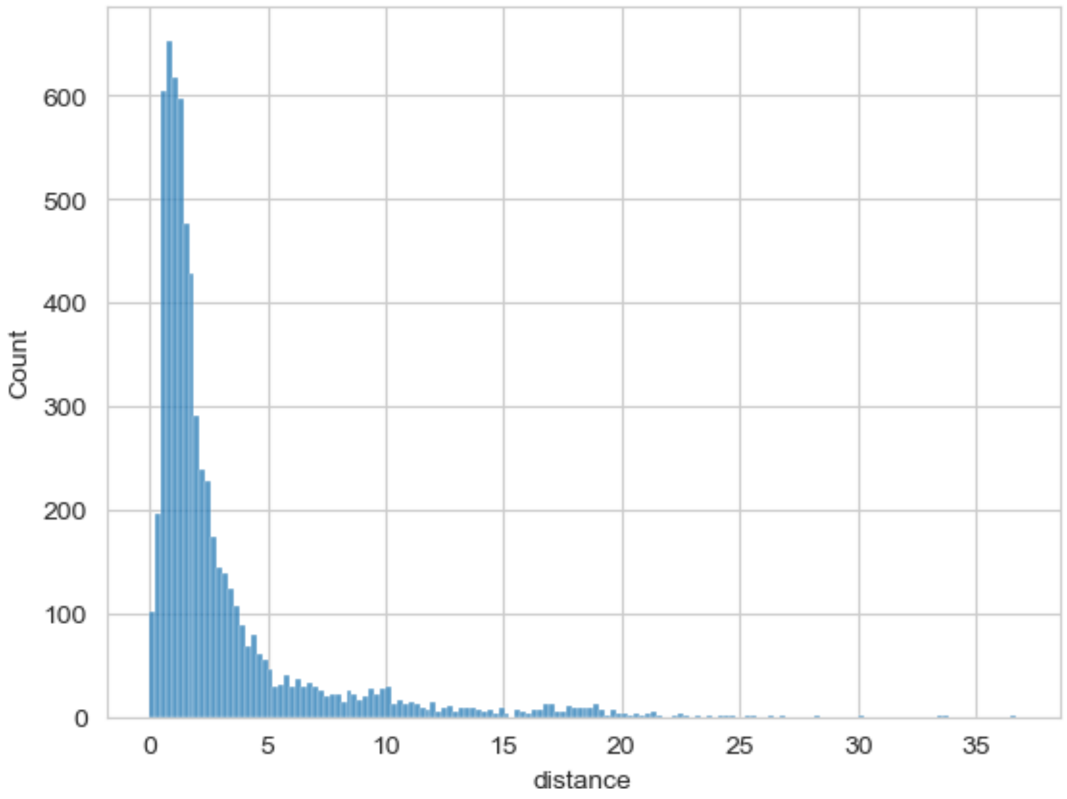
[27]:	pickup	dropoff \
count	6433	6433
mean	2019-03-16 08:31:28.514223616	2019-03-16 08:45:49.491217408
min	2019-02-28 23:29:03	2019-02-28 23:32:35
25%	2019-03-08 15:50:34	2019-03-08 16:12:51
50%	2019-03-15 21:46:58	2019-03-15 22:06:44
75%	2019-03-23 17:41:38	2019-03-23 17:51:56
max	2019-03-31 23:43:45	2019-04-01 00:13:58
std	NaN	NaN

	passengers	distance	fare	tip	tolls	total
count	6433.000000	6433.000000	6433.000000	6433.00000	6433.000000	6433.00
mean	1.539251	3.024617	13.091073	1.97922	0.325273	18.517794
min	0.000000	0.000000	1.000000	0.00000	0.000000	1.300000
25%	1.000000	0.980000	6.500000	0.00000	0.000000	10.800000
50%	1.000000	1.640000	9.500000	1.70000	0.000000	14.160000
75%	2.000000	3.210000	15.000000	2.80000	0.000000	20.300000
max	6.000000	36.700000	150.000000	33.20000	24.020000	174.820000
std	1.203768	3.827867	11.551804	2.44856	1.415267	13.815570

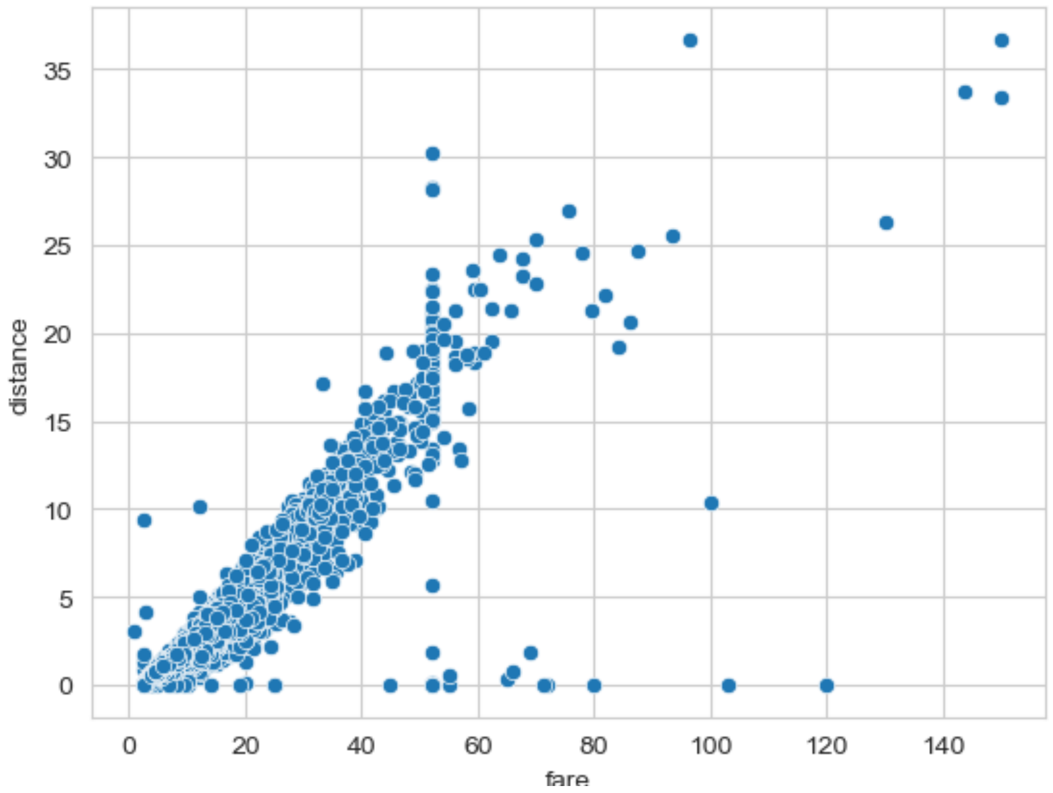
** MLO3 Comments: Using prior-knowledge of New-York taxi services, no outliers are shown to indicate there is a data quality issue. The data shows what is expected. For example, the max number of passengers is 6, which is typical capacity of a taxi.

It is helpful to do these checks visually. Histograms will help us see the distribution of a variable and scatter plots will show us the relationship between variables. Seaborn allows us to plot these using the functions histplot for histograms and scatterplot for scatter plots.

[31]:
```
sns.histplot(data=data, x='distance');
```

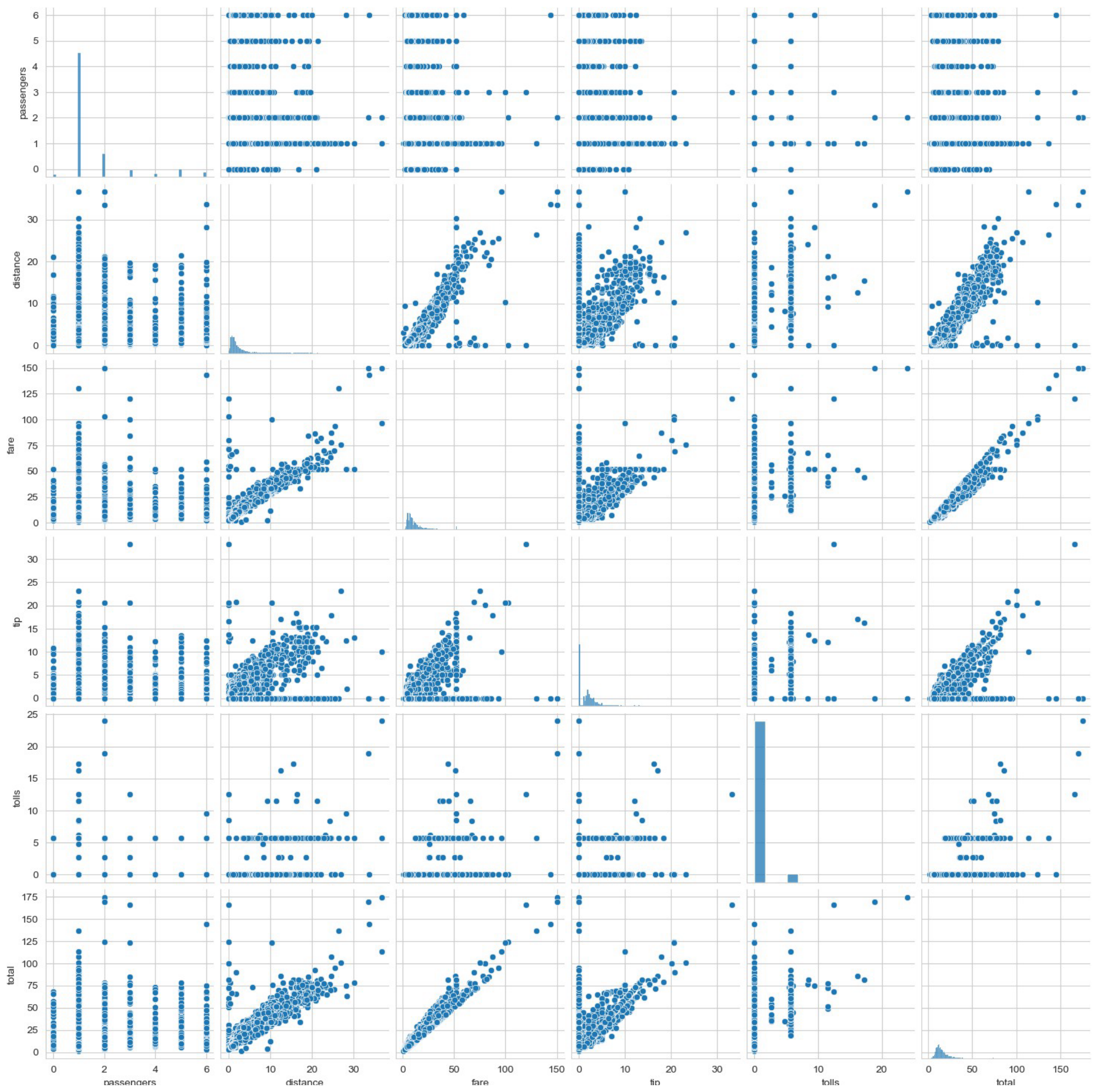


[33]:
```
sns.scatterplot(data=data, x='fare', y='distance');
```



It can be helpful to plot all the histograms and scatter plots in one go. Seaborn allows us to do this using a pairplot.

[36]:
```
sns.pairplot(data);
```



** MLO3: Using visualisations, helps to understand the strength of correaltion beteen every variable. For example, there is a strong correlation between distance and fare, which can be seen in the scatter plot. However, there is a weaker correlation between fare and tip.

The scatter plots help us to understand what kind of relationships there are between our variables and how strong they are. We can also quantify this using the Pandas method corr

[40]:
```
data.corr(numeric_only=True)
```

[40]:
```
            passengers  distance      fare       tip     tolls     total
passengers    1.000000  0.009411  0.007637  0.021099 -0.002903  0.015708
distance      0.009411  1.000000  0.920108  0.452589  0.635267  0.904676
fare          0.007637  0.920108  1.000000  0.488612  0.609307  0.974358
tip           0.021099  0.452589  0.488612  1.000000  0.413619  0.646186
```

```
tolls          -0.002903  0.635267  0.609307  0.413619  1.000000  0.683142
total           0.015708  0.904676  0.974358  0.646186  0.683142  1.000000
```

** MLO3 Comments: This table confirms what was identfied using the scatter graphs. There is a strong correlation between distance and fare (correlation is 0.92). It can be seen there is a correlation of 0.02 between tip and passengers. Therefore, I would not create a hypothesis focusing on this, instead, variables such as: distance, fare and tolls should be used.

1.1.1 Confirm your hypothesis

Now that you have explored your data and seen the relationships in it, you can pick the independent variables you want to use to predict your dependent variable. Remember, for your independent variables to be useful in predicting the dependent variable, they need to:

- influence the dependent variable
- Have a linear relationship with the dependent variable
- Have a similar distribution to the dependent variable
- Influences more the dependent variable than any of the other chosen independent variables

---

# Train and test split of data Back to Contents

We can use the scikit-learn function `train_test_split` to divide our data in two: 80% in `train_data` and 20% in `test_data`

[45]:
```
train_data, test_data = train_test_split(data,
                                         test_size= 0.2,
                                         random_state= 42)
```

# Build the model Back to Contents

Now we have our data in a form we can build our linear regression model.

Our predictor variables, the independent variables, are:

[49]:
```
independent_variable_names = ['tolls', 'distance']
```

Our predicted variable, the dependent variable, is:

[52]:
```
dependent_variable_name = 'fare'
```

We will use statsmodels Ordinary Least Squares `ols` function.

[55]:
```
independent_vars = train_data[independent_variable_names]
independent_vars = sm.add_constant(independent_vars)

data_model = sm.OLS(train_data[dependent_variable_name], independent_vars).fit()
data_model.summary()
```

[55]:

Dep. Variable:	fare	**R-squared:**	0.851
Model:	OLS	**Adj. R-squared:**	0.851
Method:	Least Squares	**F-statistic:**	1.474e+04
Date:	Fri, 15 Nov 2024	**Prob (F-statistic):**	0.00
Time:	14:26:41	**Log-Likelihood:**	-15018.
No. Observations:	5146	**AIC:**	3.004e+04
Df Residuals:	5143	**BIC:**	3.006e+04
Df Model:	2		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
const	4.7559	0.082	57.659	0.000	4.594	4.918
tolls	0.2251	0.057	3.967	0.000	0.114	0.336
distance	2.7317	0.021	129.947	0.000	2.691	2.773

Omnibus:	7480.392	**Durbin-Watson:**	2.042
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	3747184.716
Skew:	8.575	**Prob(JB):**	0.00
Kurtosis:	134.080	**Cond. No.**	6.94

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

** MLO3 Comments: This code helps identify a line of best fit. The table shows the models significance levels (0.025,0.975). The R-squared and the Adj. R-Squared have the same value of 0.851 suggesting this is a stong model with little overfitting occuring. The AIC & BIC are not useful at this point but are when comparing models.

## Interpret the model results Back to Contents

Now we have built the model we can look at the model quality and what it tells us.

- The **R squared** is a measure of how much of the variation in dependent variable is explained by the independent variables.
- The **Adj. R-squared** gives us an understanding of how much of the variation in dependent variable is explained by the independent variables. It is useful when there is more than one independent variable

The coefficients will give us the equation for our linear regression model line:

$$\hat{Y} = \text{const} + \text{coeff} \times X$$

How accurate are the estimated coefficients? Do all of our independent variables contribute to the model?

## Evaluate the model predictions using the test data Back to Contents

We can now calculate the model predictions using the `test_data`

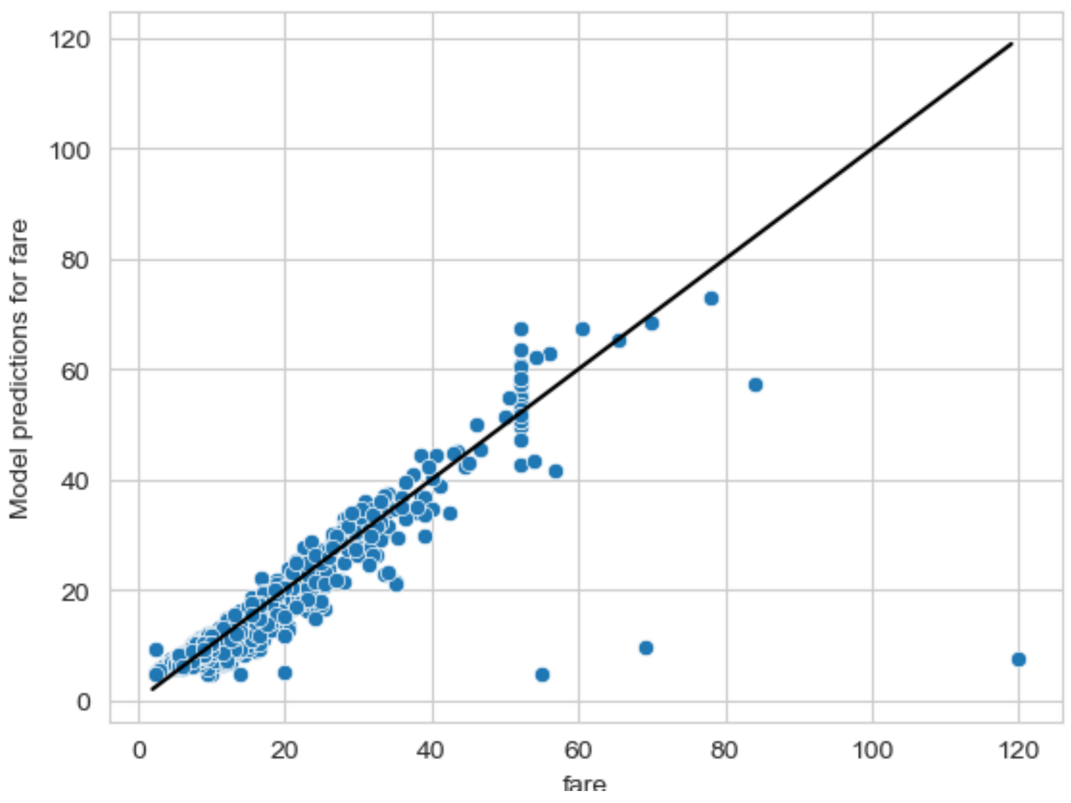
[60]:
```
predictions = data_model.predict(sm.
    ↪add_constant(test_data[independent_variable_names]))
```

A visual comparison between the predicted and actual values of the dependent variable will help us to guage the quality of the predictions. If they match, then they should all be along a straight line with a gradient of 1: as the actual values increase by 1, so too do the predictions.

[63]:
```
# Generate the scatter plot
ax = sns.scatterplot(x=test_data[dependent_variable_name], y=predictions)
ax.set_ylabel(f'Model predictions for {dependent_variable_name}')

# Find the start and end of the best prediction line
dependent_min = int(test_data[dependent_variable_name].min())
dependent_max = int(test_data[dependent_variable_name].max())

# Plot the best prediction line
sns.lineplot(x=range(dependent_min,dependent_max), y=range(dependent_min,␣
    ↪dependent_max), c='k', ax=ax);
```



We can also use these predictions to calculate the mean squared error, using `mean_squared_error`, and the r-squared, using `r2_score`. How do the R-squared values compare for the training data and the test data? Is the model likely to be useful for predicting the population?

[66]:
```
print("mean squared error \t",␣
↪mean_squared_error(test_data[dependent_variable_name], predictions))
print("R-squared model \t", data_model.rsquared)
print("R-squared predictions \t", r2_score(test_data[dependent_variable_name],␣
↪predictions))
```

```
mean squared error       21.480944664737176
R-squared model          0.8514803372231301
R-squared predictions    0.8305638692450311
```

** MLO3 Comments: The graph shows the line of best fit is accurate for the model prediction and the actual value of fare, demonstrating the model is accurate with using distance and tolls to increase the fare. Also looking at the two R-Squared values they are extremely close in values being 0.851 and 0.83, showing the model is reliable.

Bibliography

Abdullah, S. *Taxis Dataset.* Kaggle, Available at: https://www.kaggle.com/datasets/abdmental01/taxis-dataset-yellow-taxi (Accessed: 04 November 2024)

Fleming, S. *DevOps: Introduction to DevOps and its impact on Business Ecosystem,* pp 7-9 Available at: https://www.perlego.com/book/975639 (Accessed: 13 November 2024).

GO FAIR (2022) *Fair principles. GO FAIR*. Available at: https://www.go-fair.org/fair-principles/,(Accessed: 06 November 2024)

Hub, G.D.Q. (2021) *Meet the data quality dimensions, GOV.UK.* Available at: https://www.gov.uk/government/news/meet-the-data-quality-dimensions, (Accessed: 06 November 2024)

IBM (no date) *Data Lifecycle Management. IBM*. Available at: https://www.ibm.com/uk-en/topics/data-lifecycle-management, (Accessed: 06 November 2024)

IBM (no date) *What is Overfitting? IBM.* Available at: https://www.ibm.com/topics/overfitting, (Accessed 08 November 2024)

Jotform (2023) *6 Key Data Management principles, The Jotform Blog. Jotform* . Available at: https://www.jotform.com/blog/data-management-principles/, (Accessed: 08 November 2024)

Mbanje, S and Lunga, J. (2017) *Fundamental principles of supply chain management,* pp.167-168 (Accessed: 13 November 2024)

Mixson, E. (2021) *H&M restyled: Inside H&M 's Digital Transformation*. Available at: https://www.intelligentautomation.network/resiliency/articles/hm-restyled-inside-hm-digital-transformation, (Accessed: 05 November 2024)

Palacios, N. *Tech enhanced checkout for a better in store experience case study*. Medium. Available at:https://npalaciosu.medium.com/tech-enhanced-checkout-for-a-better-in-store-experience-case-study (Accessed: 03 November 2024)

Roberts, F. (2017) *Morrisons uses AI to stock its stores and drive sales, Internet of Business.*Available at: https://internetofbusiness.com/morrisons-ai-stores-sales/ (opens in a new tab) (Accessed: 07 November 2024).

Vogels, W. (2022) *5 tech predictions for 2023 and beyond, according to Amazon CTO Dr Werner Vogels.* Available at: https://www.aboutamazon.com/news/aws/werner-vogels-tech-predictions-2023 (Accessed: October, 2024)